TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto

Timo Kiravuo

Mach-käyttöjärjestelmän ja verkonhallintaprotokollien soveltuvuus automaatio- ja valvontakäyttöön

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi
diplomi-insinöörin tutkintoa varten Espoossa 19.4.1999

Työn valvoja: Professori Martti Mäntylä

Työn ohjaaja: Tekniikan tohtori Pekka Nikander

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Timo Kiravuo

Työn nimi: Mach-käyttöjärjestelmän ja verkonhallintaprotokollien soveltuvuus automaatio- ja valvontakäyttöön

Päivämäärä: 19.4.1999 **Sivumäärä:** 70

Osasto: Tietotekniikan osasto

Professuuri: Tik-86 Tietotekniikka

Työn valvoja: Professori Martti Mäntylä

Työn ohjaaja: TkT Pekka Nikander

Työssä tutkittiin mikrokernelikäyttöjärjestelmien ja verkonhallintaprotokollien soveltuvuutta automaatiokäyttöön, erityisesti tutkittiin Mach-käyttöjärjestelmää ja SNMPv1-protokollaa. Työ sisälsi sekä kirjallisuustutkimuksen että prototyyppilaitteiston suunnittelun ja toteutuksen.

Mikrokernelikäyttöjärjestelmät ovat käyttöjärjestelmiä, joissa perinteisen käyttöjärjestelmän ytimen eli kernelin toiminnoista on mahdollisimman suuri osa siirretty ytimen ulkopuolelle. Tämä mahdollistaa ytimen koon pienentämisen ja järjestelmän luotettavuuden kasvattamisen. Mach on akateemisessa maailmassa tutkimuskäyttöön kehitetty suosittu mikrokerneli.

Verkonhallintaprotokollat ovat tietoverkkojen hallintaan kehitettyjä määrämuotoista dataa siirtäviä protokollia, jotka tarjoavat hallittavan laitteen teknisestä toteutuksesta riippumattoman abstraktiokerroksen, mikä mahdollistaa tekniseltä toteutukseltaan erilaisten järjestelmien hallinnan yleiskäyttöisillä työkaluilla. Simple Network Management Protocol (SNMP) on verkonhallintaan kehitetty strukturoitua parametridataa siirtävä protokolla, joka soveltuu myös muun parametridatan siirtoon, esimerkiksi automaatiojärjestelmien valvontaan ja ohjaukseen.

Prototyyppilaitteistona toteutetun virvoitusjuoma-automaatin ohjausjärjestelmän käyttöjärjestelmänä oli alussa Mach, mutta se korvattiin teknisten vaikeuksien takia Linux-käyttöjärjestelmällä. Linuxille rakennettiin automaatin SNMP-ohjaus, jolle toteutettiin SNMP-rajapintaa käyttäen WWW- ja tekstipohjainen käyttöliittymä.

Sulautettujen järjestelmien käyttöjärjestelmänä Mach ei täyttänyt toiveita. Mach on tutkimuskäyttöjärjestelmä, jota ei tueta. Machista saatuja kokemuksia ei void kaikilta osoin soveltaa yleisemmin mikrokerneleihin.

SNMP soveltuu hyvin ei-aikakriittiseen etävalvontaan, sillä varauksella että versio 1 (SNMPv1) ei tarjoa tietoturvaa. Volyymidatan siirtoon SNMP soveltuu huonosti. TCP/IP-protokollaperheeseen perustuvana protokollana SNMP mahdollistaa yleisten tietoverkkojen käytön ja maailmanlaajuisen toimintakentän.

Hakusanat: Sulautetut järjestelmät, mikrokernelit, Mach, verkonhallinta, SNMP, kiinteistövalvonta

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author: Timo Kiravuo

Title: Using Mach operating system and network management protocols for automation

Date: 19.4.1999 **Number of pages:** 70

Department: Faculty of information technology

Professorship: Tik-86 Information technology

Supervisor: Professor Martti Mäntylä

Instructor: Dr. Sc. Pekka Nikander

This thesis deals with using microkernel operating systems and network management protocols for automation, especially using the Mach operating system and SNMPv1 protocol. Thesis includes both literature research and planning and implementing an actual prototype.

Microkernel operating systems have as much as possible of the functionality of traditional operating systems moved out from the kernel to user space. This reduces the size of the kernel itself and incresing the reliability of the system. Mach is a popular research microkernel, developed in the academic community.

Network management protocols have been developed for managing computer networks. They transport formatted data units and provide an abstraction layer, which is independent of the actual technical implementation of the device managed. This enables the management of functionally like, but technically different systems with general purpose software tools. Simple Network Management Protocol (SNMP) has been developed for computer network mangement and it can be used to transport other kinds of parameter data, like the management data for automation systems.

The prototype system, which is a soft drink machine, was first automated by using the Mach operating system. After technical difficulties Mach was replaced with the Linux operating system to support the SNMP management. Using the SNMP interface both a WWW and a text based user interface was built.

Mach is an unsupported research operating system and it did not fulfill the hopes set for an embedded systems operating system. Results from the study of Mach can not be geralized to other microkernel operating systems.

SNMP was found to fit well for the needs of non time critical automation management, with the reservation that version 1 (SNMPv1) has no data security. SNMP is not usable for the transfer of high volume data. Based on TCP/IP, SNMP can use public computer networks and is suitable for world wide operations.

Keywords: Embedded systems, microkernels, Mach, network management, SNMP, building automation

Alkulause

Tämä diplomityö on tehty Nixu Oy:lle osana yrityksen sisäistä koulutus- ja tuotekehityspanostusta.

Haluan lausua parhaimmat kiitokseni kaikille minua tämän projektin aikana tukeneille ja kannustaneille. Erityisesti haluan kiittää professori Martti Mäntylää pitkämielisyydestä ja kärsivällisyydestä, ohjaajaaniPekka Nikanderia tieteellisen haastavuuden asenteesta, Timo Pekurista ja Lauri Aarniota teknisestä avusta limuautomaatin kanssa, Jonna Partasta ja vanhempiani oikoluvusta sekä Nixu Oy:tä ja sen henkilökuntaa projektin mahdollistamisesta ja henkisestä tuesta.

Helsingissä 18.4.1999

Timo Kiravuo

SISÄLLYSLUETTELO

Lyhenteitä ja termejä

agentti	(agent) verkonvalvonnalla hallittava tietokone tai siinä oleva hallinnan mahdollistava ohjelmisto
roskankeruu	(garbage collection) ei enää käytössä olevien muistiobjektien varaaman muistin vapauttaminen
Internet	suuri internet-teknologiaan perustuva kansainvälinen tietoverkko
internet	yleisnimitys lähiverkkojen yhdistämistekniikoille, erityisesti TCP/IP-protokollaperheelle
intranet	organisaation internet-teknologioihin perustuva sisäinen verkko tai verkkopalvelu
IP	Internet Protocol, TCP/IP-protokollien alla oleva yhteydetön siirtoprotokolla
LON	Local Operating Network, eräs kenttäväyläjärjestelmä, erityisesti käytössä kiinteistöhallinnassa
hallinta-asema	(management station) kone jossa on verkonvalvonnan hallintaohjelmisto
MAP	Manufacturing Automation Protocol, tuotannonohjauksen ylemmän tason ohjausprotokolla, ei kovinkaan laajalti käytössä
MIB	Management Information Base, tietokantamäärittely, jonka valvottu laite toteuttaa
mikrokerneli	toiminnoiltaan riisuttu käyttöjärjestelmän ydin
SNMP	Simple Network Management Protocol
tehtävä	(task) Mach-käyttöjärjestelmän nimitys suoritettavalle ohjelmalle ja sen käyttämille tietorakenteille
TCP	Tranmission Control Protocol, TCP/IP-protokollaperheen yhteydellinen tiedonsiirtoprotokolla

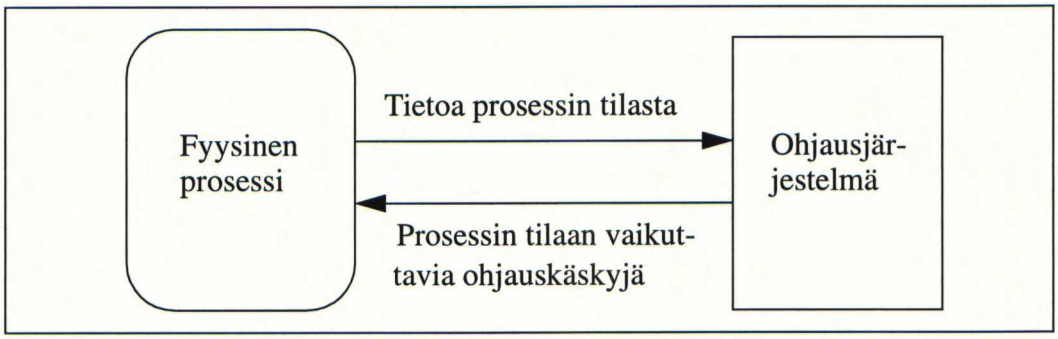
UDP	User's Datagram Protocol, TCP/IP-protokollien yhteydetön tietosähkeprotokolla

1. Johdanto

1.1. Taustaa

1.1.1. Prosessin ohjaus

Automaatio on fyysisten toimintojen toteuttamista ohjaavan logiikan sanelemalla tavalla anturoinnista saatavan mittausinformaation perusteella. Tällainen ohjauslogiikka voidaan toteuttaa eri tavoin; mekaanisesti, releillä, elektroniikalla tai mikroprosessoreilla. Joustavuutensa ja muunneltavuutensa ansiosta ohjaavan logiikan toteuttaminen mikroprosessoreilla on nykyään yleistymässä. Siinä missä mekaaninen ohjauslogiikka sidotaan jo suunnitteluvaiheessa tiettyihin tehtäviin, standardoitu prosessorilogiikka saadaan toteuttamaan erilaisia tehtäviä ohjelmoimalla se tarpeen mukaan.



Kuva 1. Prosessin ohjaus

Prosessoritekniikkaan perustuvat ohjauslogiikat voidaan jakaa karkeasti ottaen kolmeen ryhmään:

Ohjelmoitavat logiikat, jotka koostuvat prosessorista ja muistista mutta emuloivat ohjelmointirajapinnallaan perinteisiä, releistä rakennettuja logiikoita.

Matalan tason kielellä ohjelmoitavat prosessorit, joissa on minimaaliset palvelut tarjoava käyttöjärjestelmä, mutta joita käytettäessä ohjaava logiikka joudutaan toteuttamaan hyvin laitteistoläheisesti.

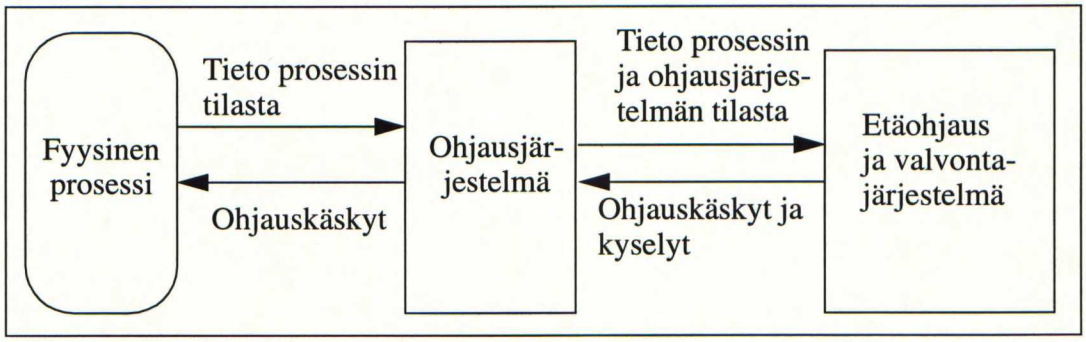
Korkeamman tason ohjelmointikielellä ohjelmoitavat käyttöjärjestelmällä varuste-

tut tietokoneet, joissa ohjaava logiikka erotetaan ohjauslaitteen fyysisistä ominaisuuksista korkeammalle abstraktiotasolle. Käyttöjärjestelmä voi olla yleiskäyttöinen tai erityisesti prosessinohjaukseen suunniteltu.

Merkittävä osa automaatiosta toteutetaan edelleenkin kahdella ensimmäisellä tekniikalla. Tällöin ympäristön alkeellisuus rajoittaa järjestelmän rakentajan mahdollisuuksia käyttää kehittyneempiä ohjausalgoritmeja, kuten neuraaliverkkoja tai sumeaa logiikkaa.

1.1.2. Prosessin etähallinta

Tietoliikenteen kehitys on heijastunut myös automaatioon. Tietoliikenne mahdollistaa järjestelmän fyysisten ulottuvuuksien kasvattamisen jopa globaaliin mittakaavaan. Tietoliikennettä tarvitaan sekä anturoinnin yhdistämiseen ohjaavaan logiikkaan että korkeamman tason ohjaus- ja valvontainformaation siirtämiseen.



Kuva 2. Prosessin etähallinta

Viime vuosina kehittyneet **kenttäväylät** mahdollistavat useiden anturien liittämisen yhteen tietoliikenneväylään. Kenttäväylät yksinkertaistavat anturoinnin kaapelointia ja syntyneet standardit yksinkertaistavat anturien konfigurointia ja ylläpitoa.

Kenttäväylät ovat kuitenkin yleensä selkeästi rajoittuneita anturointiväyliä. Rajoittavia tekijöitä ovat fyysisen kaapeloinnin pituus, anturien lukumäärä ja tiedonsiirtokapasiteetti. Ne eivät tarjoa korkeampaa abstraktiotasoa ohjattavaan järjestelmään; välitettävä data on suoraan järjestelmästä saatavaa anturointidataa.

Ohjauslogiikoiden yhdistämiseen ja järjestelmän hallitsemiseen korkeammalla abstraktiotasolla ei ole vallalla olevia standardeja. Alan kirjallisuudessa on viittauksia MAP-tietoliikennestandardiin (Manufacturing Automation Protocol), jota ei kuitenkaan käytännössä ole käytössä juuri missään. Markkinoilla oleviin ohjausjärjestelmiin voidaan yleensä muodostaa etäkäyttöyhteys, mutta tiedon siirto tapahtuu järjestelmäkohtaisten käytäntöjen mukaisesti.

Tietoliikenteen kehittyessä on tietoliikennepuolelle kehittynyt **SNMP-verkonhallintastandardi** (Simple Network Management Protocol), joka luo abstraktiotason ohjattavan järjestelmän ja sen toteutuksen välille. Ei ole mitään erityistä syytä, miksi SNMP ei soveltuisi myös muiden laitteiden kuin tietoliikennekomponenttien hallintaan.

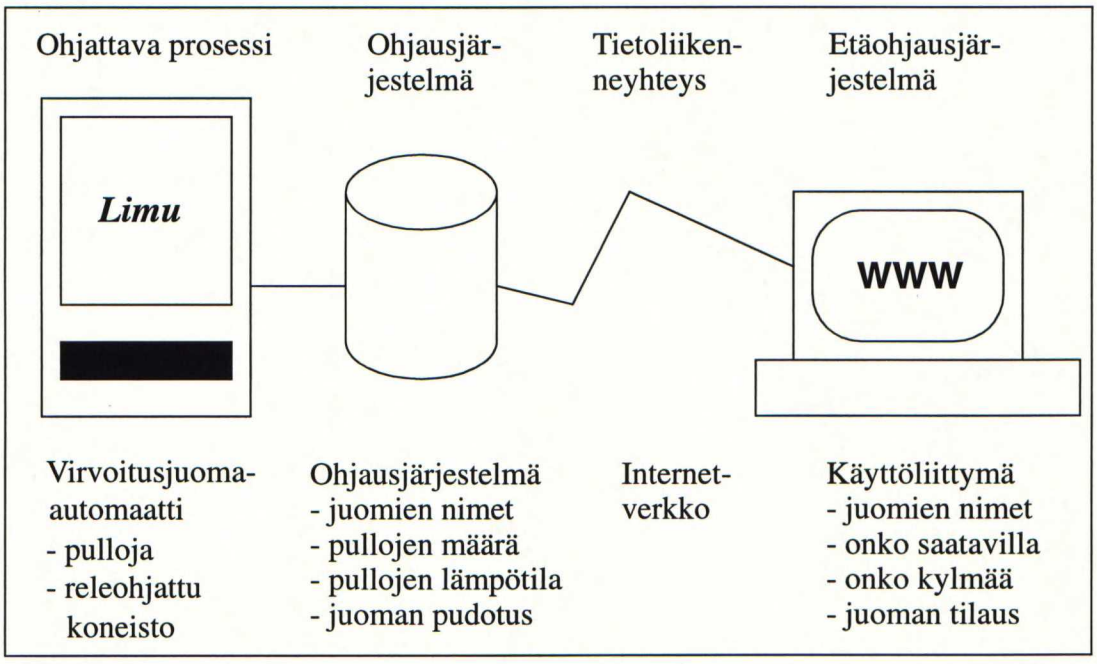
1.1.3. Modernit ratkaisut

Tieto- ja tietoliikennetekniikan kehityksen nykyisellä tasolla ohjauslogiikan pitäisi olla toteutettavissa käyttäen uudelleenkäytettäviä ohjelmistokomponentteja, laitteistoriippumattomasti ja laitteiston rajoitusten hallitsematta koko ohjelmistoprojektia.

Tätä työtä tehdessäni on mielessäni ollut ajatus järjestelmästä, jossa varsinaisen prosessin ohjaus toteutetaan yksinkertaisella ja siten luotettavalla ohjelmoitavalla logiikalla ja anturointi siihen soveltuvalla kenttäväylällä, mutta korkeamman tason ohjaus ja keskitetty valvonta toteutetaan laitteistoriippumattomien abstraktiotasojen ja standardien avulla yleiskäyttöisemmillä välineillä.

Esimerkkitoteutuksena rakennettu virvoitusjuoma-automaatin ohjaus ei eroa kovinkaan merkittävästi esimerkiksi kiinteistön etähallinnan tarpeista ja samoja hallintalaitteita- ja ohjelmistoja voitaisiin yhtenäisten standardien avulla käyttää erilaisiin käyt-

tötarpeisiin, mikä vähentäisi kehityskuluja.



Kuva 3. Työssä toteutettu virvoitusjuoma-automaatin ohjaus

Korkeamman tason hallinnan luotettavuusvaatimukset voivat olla matalammat kuin varsinaisen prosessin toiminnalle kriittisen ohjauslogiikan. Tämä mahdollistaa edullisempien järjestelmien ja tietoliikenneyhteyksien käytön. Tässä työssä olen olettanut Internet-verkon leviävän nykyisen puhelinverkon tavoin käytännössä kaikkialle ja tarjoavan siten edullisen tietoliikenneväylän.

1.2. Tavoite

Työn tavoitteena on esitellä modernia teknologiaa käyttävä julkisiin ja avoimiin standardeihin perustuva, laitteisto- ja valmistajariippumaton ratkaisu sulautettujen järjestelmien laajan mittakaavan etävalvontaan ja -ohjaukseen.

Tutkimus keskittyi yleiseen tekniseen toteutukseen ja sen tarjoamiin mahdollisuuksiin. Aiheeseen liittyy teknisen toteutuksen lisäksi myös taloudellisia, logistisia ja käytännön sovellusalueisiin liittyviä näkökulmia, joita tuodaan sopivissa määrin esille.

1.3. Tutkimusmenetelmä

Tutkimusmenetelmänä käytettiin kirjallisuustutkimusta ja prototyyppilaitteiston rakentamista. Kirjallisuustutkimuksen tavoitteena oli kartoittaa tarjoilla olevat teknologiavaihtoehdot ja niiden soveltuvuus tavoitteen toteuttamiseen.

Rakennettavalla laitteistolla pyrittiin varmistamaan kirjallisuudesta saatava informaatio sekä syventämään tutkimusta sitomalla se käytäntöön. Toteutusteknologioiksi valittiin Mach-käyttöjärjestelmä ja SNMP-verkonhallintaprotokolla. Valinnoissa painoi erityisesti se, että haluttiin suosia avoimia käyttöjärjestelmiä ja protokollia. Työn aikana Mach osoittautui liian vaikeaksi ympäristöksi, joten se korvattiin Linux-käyttöjärjestelmällä.

1.4. Rajaus

Kirjallisuustutkimuksessa pyrittiin luomaan käsitys alan kehityksestä yleisellä tasolla. Painopiste oli avoimissa käyttöjärjestelmissä ja standardoiduissa tietoliikenneratkaisuissa.

Esimerkkilaitteistona toteutettu virvoitusjuoma-automaatti on yksinkertaisena laitteena helppo ymmärtää ja sopii hyvin peruskäsitteiden esittämiseen.

Työssä toteutettu järjestelmä on toteutettavuustutkimus, ei tuotantolaitteisto. Varsinaisen tuotantolaitteiston suunnittelu ylittäisi projektin resurssit. Sarjatuotannossa laitteiston tulisi olla tuotettavissa 200 kappaleen sarjoissa alle 1000 mk:n hintaan.

Tutkimuksen painopiste on tietoliikenteessä ja etäkäytössä, sulautettujen järjestelmien ohjaus on jo tutkittua aluetta, mutta etäkäyttö ja valvonta eivät ole.

Esitetyssä ratkaisumallissa on tietoturvallisuuspuutteita, mutta tätä aluetta ei tutkita tarkemmin, koska se on ratkaistavissa muilla teknologioilla.

Automaatiojärjestelmän toimintavarmuus on tärkeä ominaisuus. Tässä työssä on oletettu alemman tason ohjausjärjestelmän tarjoavan riittävän luotettavuuden, jolloin

ylemmän tason hallintajärjestelmän luotettavuus ei ole kriittinen rajoitus järjestelmän kokonaistoiminnalle.

Tietoliikenteeseen käytettiin internet-protokollia, koska Internet-verkko on puhelinverkon ohella ainoa merkittävä maailmanlaajuisen mittakaavan tietoliikenneverkko ja kustannusrakenteeltaan selkeästi puhelinverkkoa edullisempi jatkuvaan etähallintaan. Internet-protokolliin perustuvaa tietoliikennettä voidaan hyödyntää myös puhelinverkon ylitse.

Kaikissa valinnoissa tavoitteena oli löytää julkisesti lähdekooditasolla saatavilla oleva avoin ja yleisiä standardeja noudattava ympäristö.

Käyttöjärjestelmäksi haluttiin käyttöjärjestelmä, joka pystyy toimimaan sulautetussa järjestelmässä.

1.5. Diplomityön sisältö

Tutkimuksen kohde jakautuu selkeästi kahteen toisistaan riippumattomaan osaan:

- ohjelmisto- ja laitteistoympäristöön ja
- tietoliikenteeseen.

Toisessa luvussa esitellään automaatiotekniikan taso vuoden 1995 tietojen perusteella ja ne tieto- ja tietoliikennetekniikan alueet, joita halutaan automaatiotekniikalle tarjota.

Kolmannessa luvussa pohditaan mahdollisia sovellusalueita esitettävälle ratkaisulle.

Neljännessä luvussa esitellään projektin ensimmäinen osuus ja siinä Mach-käyttöjärjestelmällä toteutettu prototyyppi.

Viidennessä luvussa kuvataan lopullinen prototyyppi ja sen SNMP-protokollalla toteutettu ohjausjärjestelmä.

Kuudennessa luvussa on työn tekemisen jälkeen vuonna 1998 tehty toinen kirjalli-

suustutkimus, jossa esitellään alan uudempia kehityssuuntia.

Seitsemännessä luvussa arvioidaan työn tuloksia, pohditaan mitä tehdystä työstä on opittu ja visioidaan tulevaisuuden mahdollisuuksia.

Lähteiden jälkeen on liitteenä työssä toteutettu MIB-tietokantamäärittely.

2. Sulautetut käyttöjärjestelmät ja tietoliikenne vuonna 1995

Tämä luku perustuu alan kirjallisuudessa, lehdistössä ja Internet-verkon tietopalvelimissa oleviin tietoihin sulautetuista käyttöjärjestelmistä ja verkonhallinnan tietoliikenteestä. Lähteet ovat vuodelta 1995 ja sitä ennen, tämä siis kuvaa tilannetta, joka oli työhön ryhdyttäessä. Kuudennessa luvussa kerrotaan miten ala on kehittynyt tämän jälkeen.

Sulautettujen järjestelmien käyttöjärjestelmistä esitellään yleiset vaatimukset ja tarkastellaan tarjoilla olevia vaihtoehtoja ja sitä, miten hyvin ne täyttävät nämä edellytykset. Tietoliikenteestä esitellään verkonhallinnan yleispiirteet ja SNMP-protokolla, sekä tarkastellaan muutamaa muuta vaihtoehtoa sulautettujen järjestelmien etähallintaan.

2.1. Sulautettujen järjestelmien käyttöjärjestelmät

Sulautetun järjestelmän käyttöjärjestelmän valintaan vaikuttaa useita eri tekijöitä. Käyttöjärjestelmän objektiivisina ominaisuuksina voidaan pitää seuraavia. [comp.realtime FAQ]

- resurssivaatimukset (muisti, CPU:n teho)
- luotettavuus
- empiirinen kokemus luotettavuudesta ja toimivuudesta
- reaaliaikaisuuden vaatimat ominaisuudet (muistialueen lukitus virtuaalimuistia vastaan, prosessin tai säikeen etuoikeus tai lukitus, mahdollisten keskeytysten hallinta, prosessien priorisointi tai muut prosessien ajoitusmekanismit)
- muistinhallinta
- liittäminen fyysisiin prosesseihin (I/O-ominaisuudet, laiteohjaimet)
- yleiset käyttöjärjestelmäpalvelut (tiedostojärjestelmä, tietoliikenne)
- ohjelmointityökalut (kääntäjät, debuggerit, kirjastot)
- suorituskyky, kuten kontekstin vaihtoaika, keskeytyksen käsittelyn aiheuttama latenssi

- dokumentaatio
- ohjelmointirajapintojen selkeys.

Koska käyttöjärjestelmä on usein sidottu tiettyyn prosessoriperheeseen ja vaikuttaa laitteistoon, on tässä yhteydessä tarkasteltava myös laitteistolle asetettavia vaatimuksia, joista merkittävimpiä ovat

- vähäinen virrankulutus
- soveltuvuus fyysiseen ympäristöön (lika, tärinä, kosteus jne.)
- fyysinen koko
- luotettavuus.

Käyttöjärjestelmän ja laitteistoarkkitehtuurin valintaan vaikuttavat myös erilaiset reaalielämän vaatimukset, historialliset ja taloudelliset tekijät, kuten

- tehdyt investoinnit (osaaminen, kehityslaitteistot, olemassa oleva ohjelmakoodipohja)
- valmistajan ja käyttäjäyhteisön tuki
- tarjoilla olevat ohjelmointityökalut (emulaattorit, debuggerit jne.) ja valmiit kirjastot
- standardointi
- legacy-vaikutukset (valmistajan koko, markkinoilla säilymisen todennäköisyys, tuki jatkossa, järjestelmän laajentamismahdollisuudet)
- laajennettavuus vaatimuksien muuttuessa
- hinta
- ohjelmoinnin helppous
- liityntöjen standardointi
- ohjelmointirajapinta
- henkilökohtaiset mieltymykset.

Optimaalista ratkaisua ei ole. Mikäli järjestelmän volyymi on suuri, esim. matkapuhelimet tai autojen ohjausjärjestelmät, on järjestelmän yksikkökustannus ratkaiseva tekijä ja kehitykseen voidaan käyttää paljonkin resursseja. Pienempivolyymisissä jär-

jestelmissä kehityskustannuksien osuus nousee merkittäväksi ja niiden vähentäminen yksikköhinnan kustannuksella on usein kannattavaa. Luotettavuus nousee hallitsevaksi tekijäksi tilanteissa, joissa itse prosessi on erityisen arvokas, esimerkiksi paperikoneen ohjauksessa tai sotilassovelluksissa.

2.1.1. Reaaliaikaisuus

Reaaliaikaisuus on ominaisuus, joka aiheuttaa jatkuvasti sekaannusta sulautetuista käyttöjärjestelmistä puhuttaessa. Sulautettujen järjestelmien ja prosessinohjauksen yhteydessä reaaliaikaisuus ei tarkoita suurta nopeutta, vaan toimimista tietyn taatun vasteajan puitteissa (teollisuusautomaatiossa millisekunteja, kiinteistöohjauksessa yleensä sekunteja). Mikäli ohjattava prosessi on esimerkiksi kynttilätehtaan stearii-nisäiliön lämmitys, ovat vasteajat tuntien luokkaa, mutta ne ovat silti reaaliaikaisia: myöhästyminen aiheuttaa tulipalon tai tehtaan pysähtymisen.

Reaaliaikaiset järjestelmät voidaan jakaa vielä "koviin" ja "pehmeisiin". Lentokoneen "Fly-by-wire"-järjestelmä on esimerkki tyypillisestä kovasta reaaliaikaisuudesta. Mikäli järjestelmä ei tuota oikeaa vastetta asetetun aikaikkunan puitteissa, seurauksena katastrofaalinen järjestelmävirhe, joka ilmenee esimerkiksi savuavana reikänä maassa. Tässä työssä toteutettu virvoitusjuoma-automaatti on esimerkki pehmeästä reaaliaikaisuudesta. Järjestelmän hidas toiminta aiheuttaa palvelutason laskun, jolla voi olla taloudellisia seurauksia, mutta joita ei kuitenkaan voida pitää katastrofaalisina.[RTMAG]

Tietoliikenteen yhteydessä sanaa reaaliaikaisuus käytetään ääni- tai kuvadatan siirtoon liittyen, jolloin sillä on kuitenkin eri merkitys. Esimerkiksi puhelinkeskustelusta voidaan hukata informaatiota välistä sen laadun heikkenemättä merkittävästi, mutta viiveet puheen siirrossa aiheuttavat huomattavaa laadun heikkenemistä.

2.1.2. Reaaliaikaisen sulautetun järjestelmän toteutusvaihtoehdot

Ohjelmoitava logiikka soveltuu yksinkertaisiin tarpeisiin, mutta ei tilanteisiin, joissa

tarvitaan korkean tason tietoliikennettä, tietokantoja, merkkijonojen käsittelyä tms.

Monoliittinen ohjelma ROM-muistissa on yksinkertaisin vaihtoehto sulautettujen järjestelmien ohjelmointiin, mutta soveltuu vain kaikkein yksinkertaisimpiin tapauksiin. Tällä tavalla kannattaa toteuttaa yksinkertainen ohjelmoitava logiikka tai jokin yksinkertainen yhtä tehtävää kerrallaan suorittava sulautettu järjestelmä, kuten taskulaskin.

Yksinkertainen keskeyttämättömän moniajon toteuttava ydin ei ole kovinkaan vaikea kirjoitettava, mutta se edellyttää, että kunkin prosessin tai säikeen on luovutettava kontrolli säännöllisin väliajoin ytimelle. Vapaaehtoinen moniajo yksinkertaistaa käyttöjärjestelmää, mutta siirtää kuorman sovellusohjelmille, joiden on vapautettava prosessorin hallinta. Ohjelmiston koon kasvaessa tällainen ydin aiheuttaa ongelmia sovelluskehitykselle.

Keskeyttävä ydin siirtää vastuun reaaliaikaisuudesta osittain sovellusohjelmoijalta käyttöjärjestelmän ohjelmoijalle. Keskeyttävä ydin myöskin suojaa osittain yhden osajärjestelmän toimintavirhettä vastaan, keskeyttämättömän ytimen kohdalla yksi tehtävä pystyy lukitsemaan koko järjestelmän.

Ytimen koko

Ydin voi olla joko suuri tai pieni. Perinteisten yleiskäyttöjärjestelmien, kuten Unixin tai Windowsin ydin on varsin suuri, sisältäen erilaisia toimintoja kuten tiedostojärjestelmän. Sulautetuissa järjestelmissä muisti on usein rajoitettu resurssi. Sulautetun järjestelmän ytimen kokoa voidaan pienentää siirtämällä toimintoja ytimen ulkopuolelle alijärjestelmiin. Tällaista ydintä kutsutaan mikrokerneliksi. Esimerkiksi tiedosto- tai tietoliikennejärjestelmä voidaan siirtää ytimen ulkopuolelle käyttäjätilaan, jolloin ne voidaan vaihtaa kooltaan pienempiin ja ominaisuuksiltaan vaatimattomampiin ja jolloin niiden mahdolliset virheet eivät vaikuta itse ytimen toimintaan, mikä lisää järjestelmän vikasietoisuutta. Lisäksi resurssienkäyttöä voidaan hallita paremmin kuin ytimen sisällä, ja alijärjestelmät voidaan vaihtaa helpommin toisiin tai jopa jättää pois

tarpeettomina.

2.1.3. Ohjelmoitavat logiikat

Teollisuusautomaatio alkoi mekaanisena automaationa, josta siirryttiin releohjattuun sähköiseen automaatioon. Mikroprosessorien yleistyessä 1970-luvulla releohjausjärjestelmiä ryhdyttiin korvaamaan ohjelmoitavilla logiikoilla, jotka ovat mikroprosessoriohjattuja, releitä emuloivia järjestelmiä. Tyypillisessä ohjelmoitavassa logiikassa mikroprosessori suorittaa tiukkaa silmukkaa tuhansia kertoja sekunnissa, lukien syötelinjat ja ohjaten lähtöjä kuten vastaava releistä rakennettu logiikka.

Ohjelmoitavien logiikoiden perusominaisuuksiin kuuluu yksinkertaisuudesta johtuva suuri luotettavuus muihin prosessoripohjaisiin ohjausjärjestelmiin verrattuna ja fyysinen soveltuvuus teollisuusympäristöön.

Ohjelmoitavia logiikoita voidaan ohjelmoida eri tavoin. Ohjelmoitavan logiikan ohjelmointirajapinta perustuu kuitenkin aina fyysisten releiden simuloinnille ja niistä rakennetun verkon mallintamiselle.

Ohjelmoitavien logiikoiden tietoliikenneominaisuudet ovat yleensä varsin vaatimattomat. Logiikoissa on usein RS-232 tai Ethernet-liittymä, mutta käytettävät protokollat ovat valmistajakohtaisia ja ominaisuuksiltaan rajoittuneita.

2.1.4. Mikrotietokoneiden yleiskäyttöjärjestelmät

Automaation rakentaminen jotakin mikrotietokoneiden yleiskäyttöjärjestelmää käyttäen on mahdollista, mutta ei aina kannattavaa. Yksinkertaiset levykäyttöjärjestelmät (MS-DOS, CP/M) eivät yleensä tarjoa muita merkittäviä palveluita kuin tiedostojärjestelmän, jonkinasteista tietoliikennettä ja reaaliaikakellon. Moniajon ja käyttöjärjestelmän tukipalveluiden puuttuminen aiheuttaa sen, että käytännössä ohjausjärjestelmän on oltava kaiken ohjaukseen tarvittavan sisältävä monoliittinen ohjelma. Etenkin reaaliaikaisten järjestelmien toteuttaminen täten on työlästä. Käytännössä reaaliaikaisen ohjausjärjestelmän toteuttaminen levykäyttöjärjestelmän päälle saattaa edellyttää

oman pienen käyttöjärjestelmän kirjoittamista ja tällöinkin allaolevan käyttöjärjestelmän systeemikutsuja on vältettävä reaaliaikaisten toimintojen aikana. Mikrotietokoneiden yleiskäyttöjärjestelmien luotettavuus ei myöskään ole aina riittävä automaation tarpeisiin.

2.1.5. QNX

QNX on varsin suosittu sulautettujen järjestelmien reaaliaikainen käyttöjärjestelmä ja sopiva edustamaan omaa lajiaan. [comp.realtime]

QNX-ympäristössä ohjelmistokehitys voidaan tehdä joko QNX-käyttöjärjestelmässä itsessään tai Windows-ympäristössä. Ohjelmointirajapinnaltaan QNX muistuttaa Unixia, vaikka ei olekaan täysin Unix-yhteensopiva ympäristö.

QNX-käyttöjärjestelmästä on saatavilla kaksi versiota, QNX RTOS -käyttöjärjestelmä ja QNX Neutrino mikrokerneli.

QNX RTOS on sulautettujen järjestelmien kehittäjien piirissä suosittu i386-arkkitehtuuriin sidottu reaaliaikainen mikrokernelikäyttöjärjestelmä. Se on modulaarinen ja konfiguroitavissa tarpeen mukaan pieneen kokoon.

QNX Neutrino on sulautettuihin järjestelmiin tarkoitettu reaaliaikainen mikrokerneli, joka ei ole täydellinen käyttöjärjestelmä. Neutrino tarvitsee ROM-muistia alle 64 ktavua ja RAM-muistia alle 32 ktavua.

QNX on ollut markkinoilla varsin pitkään. Sen ympärille on kehittynyt hyvä valikoima työkaluja ja laajennuksia. Esimerkiksi graafinen käyttöliittymä voidaan rakentaa sulautettuihin järjestelmiin Photon microGUI -tuotteella tai käyttämällä QNX:lle siirrettyä X Window Systemiä.

Ohjelmoijan näkökulmasta QNX on POSIX/Unix-rajapinta mikrokernelin päällä. Mikrokerneli on aito mikrokerneli, joka toteuttaa prosessien välisen kommunikoinnin, signaalit ja laitteistokeskeytyksien käsittelyn. Muut toiminnot, jopa laiteohjaimet,

ovat mikrokernelin ulkopuolella käyttäjäntilassa.

Tässä työssä QNX:n käyttöä rajoitti sen sitoutuneisuus Intelin i386-arkkitehtuuriin ja se että QNX:n lähdekoodi ei ole saatavilla.

2.1.6. Mach

Mach-käyttöjärjestelmän kehitys alkoi 80-luvun puolivälissä Carnegie Mellon Universityssä (CMU). Tutkimusohjelman tavoitteena oli kehittää käyttöjärjestelmä, joka tekisi mahdolliseksi laitteistotekniikan uusimman kehityksen hyödyntämisen ja samalla kääntäisi ytimien koon kasvutrendin, pyrkien vähentämään ytimen toimintoja ja kokoa. Ensi alkuun tavoitteena oli vain uuden ytimen kehittäminen Unix-käyttöjärjestelmälle, mutta kehitystyön kuluessa Mach-ydin itsenäistyi omaksi ytimekseen ja sen päällä oleva Unix omaksi alijärjestelmäkseen. [Boykin93]

Machia kehitettiin CMU:ssa 1985-1994, jonka jälkeen Machin kehitys jatkui Utahin yliopistossa. Viimeinen versio Utahista on keväältä 1996, jolloin Utah ilmoitti luopuvansa Machin kehityksestä ja siirtyvänsä tutkimaan käyttöjärjestelmiä Fluke-nimisen mikrokernelin pohjalta.

Machin kehityksen tavoitteena oli korvata BSD 4.3 -Unixin ydin Mach-kernelillä ja perinteisen Unix-ytimen tehtäviä hoitavilla prosesseilla. Ajan myötä Mach-ydin alettiin kuitenkin nähdä omana käyttöjärjestelmänään, jolle Unix-käyttöjärjestelmän tukeminen oli vain yksi mahdollisista tehtävistä. Sittemmin mm. MS-DOS on toteutettu Machille ja Mach-ydin pystyy tukemaan useiden samanaikaisten käyttöjärjestelmien toimintaa. Mach on käytössä OSF/1-käyttöjärjestelmän ytimenä.

On huomattavaa että Mach-käyttöjärjestelmä tarjoaa itsessään vain mikrokernelin, joka sisältää laiteohjaimia ja prosessien välisen kommunikoinnin. Toisin sanoen pelkällä Mach-ytimellä ei tee mitään.

Machin rakenne

Mach-ympäristössä prosessi koostuu kahdesta osasta. **Tehtävä** on prosessin omistamista resursseista, kuten muistista ja Mach-porteista koostuva passiivinen kokonaisuus. **Säie** on tehtävään kuuluva suoritettava ohjelma. Samanaikaisessa ajossa voi olla useampia säikeitä yhden tehtävän sisällä.

Kommunikointi tehtävien välillä ja ytimen kanssa tapahtuu porttien ja viestien kautta. Portti on tehtävän omistuksessa oleva, ytimeen liittyvä tietorakenne ja tehtävään kuuluvat säikeet saavat käyttää ko. porttia, lähettäen viestejä ja vastaanottaen niitä. (Machin portti on täysin eri asia kuin TCP/IP:n portti.)

Mach käsittelee muistia muistiobjekteina. Tämä abstraktiotaso mahdollistaa mm. muistinhallinnan siirtämisen osittain ytimen ulkopuolelle.

RT-Mach

CMU:n Real-Time Mach on Mach-käyttöjärjestelmän reaaliaikaisia ominaisuuksia tarjoava versio, jonka kehitys päättyi Mach-kehityksen siirtyessä CMU:sta Utahiin. Tutkimuskäyttöjärjestelmänä RT-Machin ominaisuudet ovat monipuolisia, se sisältää mm. puolen tusinaa eri skedulointialgoritmia, koska kaikkien käyttäytymistä on haluttu tutkia jossain projektin vaiheessa.

2.1.7. Linux

Linux on Linus Torvaldsin liikkeelle saattaman ja Internet-yhteisössä jatketun projektin tuloksena syntynyt vapaassa jakelussa oleva Unix-yhteensopiva käyttöjärjestelmä. Johtuen avoimuudestaan ja projektin saamasta suosiosta Linux on varsin yleinen palvelinkäyttöjärjestelmä.

Sulautettuihin järjestelmiin Linuxin voidaan katsoa sopivan rajoitetusti. Suurin osa näistä rajoituksista pätee myös muihin Unix-käyttöjärjestelmiin. Peruspuutteet ovat:

- reaaliaikaisten ominaisuuksien puute

- käyttöjärjestelmän ja ytimen suuri koko, palveluita karsimallakin Linux vaatii noin yhden levykkeen (1,4 Mt) verran levymuistia ja yli megatavun keskusmuistia
- luotettavuus on toimistopalvelinkäyttöjärjestelmän luokkaa, mikä ei riitä kaikkiin teollisuuden ohjaus- ja valvontasovelluksiin.

RT-Linux

Linuxista on olemassa versio, johon on lisätty reaaliaikaisuuden vaatimia ominaisuuksia. Lähinnä tällaisia ovat muistialueiden lukitus siten, että ne eivät vaihdu levylle ja prosessin lukitus siten, että käyttöjärjestelmä ei pääse vaihtamaan välillä toista prosessia ajoon. Nämä ominaisuudet ovat kuitenkin rajoittuneita, eikä voida sanoa että RT-Linux tukisi aidosti reaaliaikaisia prosesseja, esimerkiksi prosessien prioritisointi puuttuu.

RT-Linux toteuttaa reaaliaikaominaisuudet sijoittamalla Linux-ytimen ja laitteiston väliin pienen reaaliaikaytimen, joka emuloi laitteistotasoa Linuxille ja suorittaa Linuxia alimman tason prioriteetin prosessinaan. RT-Linuxin ydin on modulaarinen ja esimerkiksi skeduleri voidaan vaihtaa. Peruskonfiguraatio on varsin vaatimaton: muisti on allokoitava prosessille staattisesti eikä muistinhallinta suojaa muuta muistia, prosessit kommunikoivat ainoastaan jaetun muistin kautta ja skeduleri ei suojaa mahdottomia suoritusvaateita vastaan. RT-Linuxissa reaaliaikainen prosessi ei siis voi käyttää varsinaisen käyttöjärjestelmän palveluita, kuten tietoliikennettä tai tiedostojärjestelmää. Kommunikointi käyttöjärjestelmän kautta on järjestettävä siten, että reaaliaikainen prosessi kommunikoi jaetun muistin tai FIFO-putken kautta Linux-prosessille, joka sitten hyödyntää käyttöjärjestelmän palveluita. Linuxin RT-ydin ei estä reaaliaikaisia prosesseja varaamasta kaikkea keskusyksikköaikaa, jolloin käyttöjärjestelmän muu toiminta pysähtyy.

Reaaliaikaiset prosessit suoritetaan Linuxin ytimen oikeuksilla ja ytimen muistialueessa, jolloin prosessin virheet voivat kaataa koko käyttöjärjestelmän.

2.1.8. Yhteenveto

Ohjelmoitavat logiikat ja alkeellisemmat yleiskäyttöjärjestelmät ovat liian rajoittuneita tähän työhön. Työn kannalta kiinnostavina sulautetun järjestelmän toteutusympäristöinä pidän seuraavia:

QNX on reaaliaikainen ydin, jonka ympärille on rakennettu käyttöjärjestelmä ja palvelut. Se on tuettu ja yleisesti käytetty ympäristö, jota voidaan pitää luotettavana. Ainoa ongelma on (tämän työn kannalta) kaupallisuus ja se, että lähdekoodi ei ole saatavilla.

Mach vaikuttaa hyvin lupavalta ympäristöltä, vaikka sen ydin ei olekaan kooltaan kovinkaan pieni (liki yksi megatavu).

Linux on perinteinen yleiskäyttöjärjestelmä kaikkine lisukkeineen ja sisältää paljon sulautettujen järjestelmien kannalta ylimääräisiä ominaisuuksia. Rt-Linux toteuttaa reaaliaikaisen käyttöjärjestelmän lisäämällä olemassa olevaan käyttöjärjestelmään reaaliaikaisia ominaisuuksia. Se sopii tilanteeseen, jossa tarvitaan kohtuullisen pieni reaaliaikainen osuus laajempaan ohjelmakokonaisuuteen.

2.2. Etäkäytön tietoliikenne

Suuremmat tietoliikenneverkot, kuten puhelinverkot ja internet-verkot ovat yleensä jatkuvan muutoksen tilassa. Verkkojen käyttömäärän muuttuessa on verkkoa muutettava vastaavasti, jolloin verkon rakenne muuttuu ja verkon laitteet (puhelinkeskukset, reitittimet tms.) on konfiguroitava uudelleen. Verkon toimintaa on myös valvottava vikatilanteiden varalta, ja verkon käytöstä kerättävä tilastointia kapasiteetin hyödyntämisen tehostamiseksi ja ongelmien ennakoimiseksi.

Näitä tarpeita varten on kehitetty erilaisia verkonhallintamenetelmiä. Tästä tutkimuksesta olen rajannut pois valmistajakohtaiset suljetut järjestelmät, jolloin jäljelle jää kaksi avointa ja yleiskäyttöistä verkonhallintaprotokollaa: SNMP ja CMIP. SNMP:tä käytetään yleisesti internet-tyyppisten verkkojen verkonhallintaan, CMIP:tä käytetään

puolestaan televerkkojen hallintaan.

2.2.1. Verkonhallinnan käsitteet

Agentti - hallinnan kohteena oleva laite tai siinä oleva agenttiohjelmisto, joka vastaa hallinta-aseman kyselyihin. Agenttiohjelmisto hakee tietoa valvottavan järjestelmän omista tietorakenteista ja esittää sen MIB-määrittelyjen mukaisessa muodossa. Agentti yleensä myös sallii valvottavan järjestelmän tietojen muuttamisen ja lähettää tarvittaessa omatoimisesti hallinta-asemalle tietoa poikkeustilanteista.

Hallinta-asema, management station, network management station - , pitää säännöllisesti yhteyttä yhteen tai useampaan agenttiin. Verkonhallinnassa hallintaohjelmisto esittää tyypillisesti verkon graafisessa muodossa kuvaten verkon tilaa erilaisin värein ja symbolein.

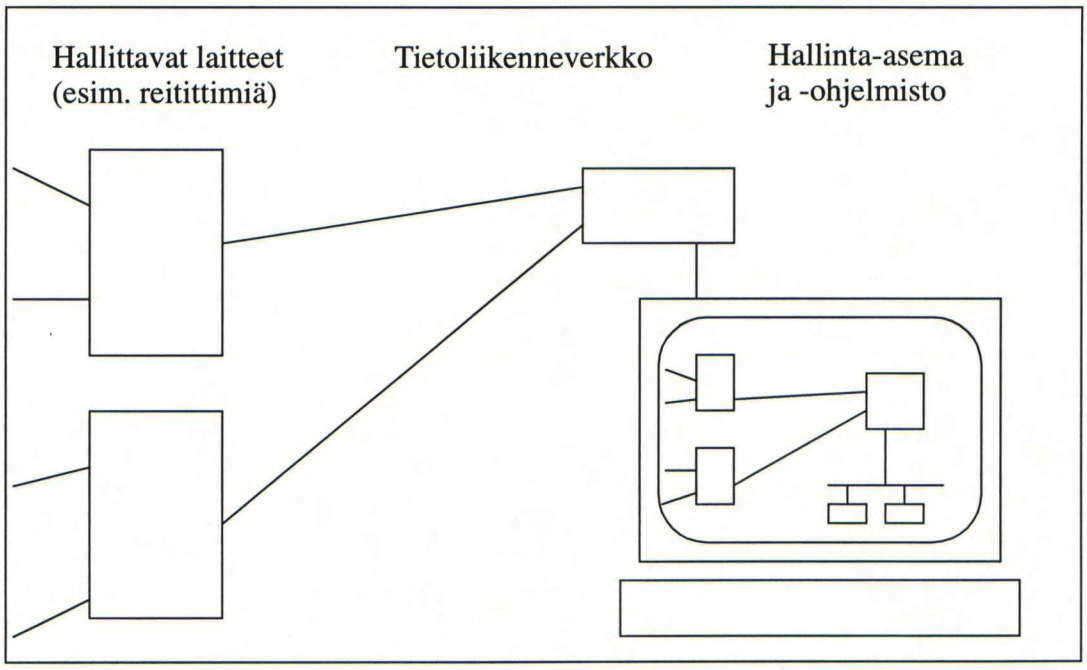
Management Information Base, MIB - termi, jolla on useampi merkitys etenkin SNMP:n yhteydessä. MIB voi tarkoittaa tietokannan hierarkkista osoiteavaruutta, tiettyä alijoukkoa tästä tietokannasta, agentissa olevia tietoja tai formaalia määrittelyä tietokannan sisällöstä. Alan kirjallisuudessa ilmenee yleensä aiheyhteydestä mitä milloinkin termillä tarkoitetaan.

2.2.2. Internet-protokollat

Internet-protokollat ovat lähiverkkoja yhdistäviä protokollia. Toimiakseen ne tarvitsevat alleen jonkinlaisen tietoliikennepaketteja välittävän kerroksen (OSI-mallin kerrokset 1 ja 2), kuten lähiverkon (esim. Ethernet), sarjayhteyden päällä paketteja välittävän protokollan (esim. PPP, Point to Point Protocol) tai muun paketteja välittävän yhteyden (esim. ATM tai frame relay).

IP (Internet Protocol)

IP-protokolla välittää yksinkertaisia datagrammeja eli tietosähkeitä tietoverkon laitteiden välillä. IP-protokolla on OSI-mallin 3. tason eli verkkotason protokolla.



Kuva 4. Verkonhallinta

SNMP, NFS	DNS	FTP, HTTP, SMTP jne.
UDP	TCP	
IP		
Ethernet-protokolla (IEEE 802.2), PPP, ATM, X.25, frame relay, tms.		
Ethernet-kaapelointi, sarjayhteys, modeemiyhteys, valokuitu tms.		

Kuva 5. TCP/IP-protokollapino

IP-paketti sisältää vastaanottavan koneen ja paketin lähettäneen koneen IP-osoitteen. IP-osoite on nykyisessä 4-versiossa 32 bittiä pitkä numeerinen osoite, tulossa olevassa versiossa 6 osoite tulee olemaan 128 bittinen. Verkkosegmenttejä yhdistävät reitittimet välittävät paketin vastaanottavaan koneeseen tämän numeerisen osoitteen perusteella.

Internet-verkoissa koneille voidaan antaa myös tekstuaalisia nimiä, käyttäen DNS-

palvelua (Domain Name System). DNS-perustuu hierarkkiseen hajautettuun tietokantaan, joka toteuttaa relaatiot nimistä IP-osoitteiksi (esim. www.nixu.fi -> 194.187.122.20) ja osoitteista nimiksi (esim. 194.187.122.20 -> www.nixu.fi)

IP-paketissa on lähettävän ja vastaanottavan koneen IP-osoite, muita vähemmän tärkeitä otsikkotietoja ja data-alue. IP-protokollan tasolla viestissä ei ole tarkempaa tietoa jakelusta vastaanottavan koneen sisällä.

IP-protokolla ei ole yhteydellinen protokolla eikä paketin perille menoa varmisteta mitenkään. IP-paketit saattavat matkalla kadota, duplikoitua, kulkea satunnaisia reittejä ja tulla perille eri järjestyksessä kuin ovat lähteneet.

UDP (User's Datagram Protocol)

UDP on IP:n päälle rakennettu yksinkertainen tietosähkeprotokolla, joka lisää IP:hen lähinnä lähettäjän ja vastaanottajan käyttämän tietoliikenneportin osoitteen. Kun pelkkä IP-protokolla välittää dataa koneiden välillä, voidaan portti-abstraktiota käyttäen liikenne välittää oikealle prosessille koneessa.

Standardoiduilla palveluilla, kuten SNMP:llä, on sovitut porttiosoitteet. SNMP-agentti on oletusarvoisesti osoitteessa 161 ja hallintaohjelmisto ottaa agenttien omatoimisesti lähettämät ilmoitukset eli trapit vastaan portissa 162.

Kohtuullisen yksinkertaisena protokollana UDP ja sen vaatima IP ovat toteutettavissa järjestelmiin, joissa on vaatimatonkin suorituskyky [Aarnio94].

TCP (Transmission Control Protocol)

TCP-protokolla toteuttaa virheettömän tietovuon IP-protokollan päälle. TCP-yhteyden luomiseen vaaditaan kolme ja purkamiseen neljä IP-pakettia, lisäksi vuonvalvonnan kuittaukset aiheuttavat kuormaa. TCP-protokolla osaa tunnistaa IP-pakettien katoamiset ja muut virheet ja pystyy toipumaan niistä omatoimisesti pyytämällä uudelleenlähetystä.

Ohjelmoijalle TCP-yhteys näkyy avauksen jälkeen putkena, johon kirjoitetaan ja jota luetaan kuten suorasaantitiedostoa. TCP-yhteys muodostetaan aina koneiden porttien välille. TCP-porttien numeroavaruus on riippumaton UDP:n porttien numeroavaruudesta.

TCP:tä ei yleensä ole käytetty verkonhallintaan. Verkonhallinnassa TCP:n etuna olisi läpinäkyvyys ohjelmoijalle, luotettavuus ja tehokas suurten tietomäärien siirto. Näiden ominaisuuksien toteuttamisen aiheuttamat vaatimukset ovat kuitenkin myös TCP:n ongelma. Vuonohjaus ja virheiden tunnistaminen edellyttävät, että TCP-toteutus on merkittävästi monimutkaisempi kuin UDP-toteutus, vaatien sekä muistia että keskusyksikön tehoa enemmän.

Erityisesti TCP aiheuttaisi ongelmia tilanteessa, jossa yhteys useaan valvottavaan koneeseen katoaa, esim. reitittimen toiminnan häiriytyessä. Tällöin valvovan koneen TCP-toteutus suorittaisi uudelleenyrityksiä käyttöjärjestelmätasolla, ja vasta time-outtien jälkeen saisi hallintaohjelmisto tiedon ongelmatilanteesta.

2.2.3. SNMP

Simple Network Management Protocol on tarkoitettu alunperin internet-tyyppisen tietoliikenneverkon ja sen komponenttien valvontaan ja hallintaan. Protokollaa käytetään kuitenkin yhä yleisemmin myös muiden verkossa olevien laitteiden ja ohjelmistojen valvontaan, hallintaan ja konfigurointiin.

SNMP:n tarjoama abstraktiokerros perustuu MIB-määrittelyyn (Management Information Base), jonka mukaan hallinta-asema pyytää hallittavalta laitteelta tietoa. SNMP-protokolla siirtää varsin elementaarisia tietoalkioita, tyypillisesti lukuja ja merkkijonoja. Jotta siirretylle datalle saataisiin merkitys, on hallintaohjelmistolle kerrottava mitä missäkin osoitteessa olevat tietoalkiot merkitsevät (esim. onko luku 7 jäljellä olevien pullojen määrä, tänään pudotettujen pullojen määrä vai ohjelmoijan onnenluku).

MIB-osoiteavarudessa jokaisella SNMP-järjestelmän tieto-objektilla on uniikki osoite. Esimerkiksi 1.3.6.1.2.1.1.4 eli iso.org.dod.internet.mgmt.mib-2.system.sysContact on kenttä, joka sisältää valvottavan järjestelmän yhteyshenkilön yhteystiedot. Koska liki jokainen SNMP-agentti toteuttaa tämän osan MIB-määrittelyistä, voi siis useimpien SNMP-valvottujen järjestelmien yhteyshenkilön saada selville lähettämällä ko. järjestelmälle kyselyn alkion 1.3.6.1.2.1.1.4 sisällöstä.

Koska MIB-osoitepuu on hierarkkinen, siitä voidaan varata haaroja eri organisaatioille, esimerkiksi 1.3.6.1.4.1.1625 eli iso.org.dod.internet.private.enterprises.nixu on oma IANA:n (Internet Assaigned Numbers Authority) Nixu Oy:lle varaama haara. Nixu Oy:n kehittämät MIB-määrittelyt (kuten virvoitusjuomalaitteen hallintaan tarvittavat tietoalkiot) tulevat tämän osoitteen alle, mikäli niitä ei standardoida yleisempään käyttöön.

MIB tietokannan osoitteilla ei ole mitään tekemistä IP-osoitteiden tai Domain Name System -nimipalvelun kanssa, tietystä yhdennäköisyydestä huolimatta.

Haaran 1.3.6.1.2 eli iso.org.dod.internet.mgmt alla ovat Internet Architecture Boardin hyväksymät yleiset Internet-objektien MIB:t, MIB-I ja sitä laajentava MIB-II [RFC 1158]. Usein termiä MIB käytetään viittaamaan juuri näihin tiettyihin tietokantoihin.

Jokainen SNMP-agentti toteuttaa jonkin osan koko mahdollisesta MIB-avaruudesta, vastaten sille tuleviin kyselyihin tiettyjen MIB-tietokantaosoitteiden kohdalta. Tyypillisesti agentti toteuttaa MIB-I:n tai MIB-II:n sekä jotain tuote- tai valmistajakohtaisia tietokantoja (esim. Nixu Oy:n LIMU-MIB:n).

Agentin toteuttama MIB esitellään formaalina määrittelynä [RFC1157] hallintaohjelmistolle, alla esimerkiksi edellä mainitun sysContact-alkion määrittely, joka on osa laajempaa MIB-II -määrittelyä.

```
sysContact OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..255))
    ACCESS read-write
```

```
        STATUS mandatory
        DESCRIPTION
            "The textual identification of the con-
    tact person for this managed node, together with
    information on how to contact this person."
        ::= { system 4 }
```

SNMP:n tietomallissa on olennaista että MIB tarjoaa abstraktiotason valvottavaan järjestelmään. SNMP:n välittämien ja MIB:n kuvaamien tietojen ei tarvitse vastata suoraan järjestelmän sisäistä rakennetta. Esimerkiksi toteutetussa virvoitusjuoma-automaatissa ei ole anturia, joka kertoisi linjassa olevien pullojen määrän, vaan SNMP-agentti generoi tämän tiedon omasta tietokannastaan, johon päivitetään pullojen määrä täytön yhteydessä ja josta vähennetään pudotetut pullot.

SNMP:n tietoliikennemalli

SNMP on siis IP- ja UDP-protokollien päälle rakennettu sovellustason protokolla. SNMPv1 tarjoaa viisi perusviestiä. [Stallings96]

GetRequest pyytää agenttia palauttamaan tietyssä MIB-osoitteessa olevan datan. Pyynnön lähettäjä ja vastaanottaja tunnistetaan IP- ja UDP-paketeissa olevien IP-osoitteiden ja porttinumeroiden perusteella.

GetResponse on agentin hallinta-asemalle lähettämä vastaus, joka sisältää tietoalkion MIB-osoitteen ja sisällön tai virheilmoituksen ASN.1-koodattuna.

GetNextRequest pyytää agenttia palauttamaan tiettyä MIB-osoitetta järjestyksessä seuraavan tietoalkion datan. Tässä hyödynnetään MIB-osoitehierarkian leksikaalista järjestystä. GetNextRequest on hyödyllinen esim. taulukkoja läpikäytäessä.

SetRequest asettaa agentissa olevan tietoalkion arvon.

Trap on agentin keino lähettää oma-aloitteisesti dataa hallintaohjelmistolle. Tyypilli-

sesti Trap-viesti lähetetään ilmoittamaan jostain poikkeustapahtumasta. Koska IP ja UDP eivät takaa tietosähkeen perillemenoa eikä protokollassa ole kuittausta Trap-viesteille, ei tämä toiminto ole luotettava.

Koska Trap-viestit eivät ole luotettavia, on SNMP käytännössä pollaava protokolla, jossa hallintaohjelmisto seuraa valvottavan laitteen tilaa.

SNMP Community

SNMP tukee hallinnoitavien laitteiden jakamista ryhmiin. Agentti voi kuulua yhteen tai useampaan ryhmään ja antaa eri ryhmille eri oikeuksia MIB-tietokantaansa. Tyypillisesti esim. public-ryhmällä on vain oikeus Get-pyyntöihin, mutta private-ryhmällä on sekä oikeus myös Set-pyyntöihin. Samoin näkymää MIB-avaruuteen voidaan rajoittaa. SNMP-viesteissä välitetään hallintaohjelman edustaman ryhmän nimi, "Community Name".

Käytännössä SNMP:n "Community Name" on selväkielinen salasana eikä se täytä avoimien tietoverkkojen tietoturvalle asetettavia vaatimuksia.

Hallintaohjelmistot

SNMP:n perinteinen käyttöalue on ollut verkonhallinta. Tämä edellyttää hallintaohjelmistolta seuraavia ominaisuuksia:

- verkon topologian dynaamista oppimista verkon rakenteen muuttuessa
- mahdollisuutta hälyttää erilaisten sääntöjen mukaan (laite ei vastaa, raja-arvo ylittyy)
- raportointia

Tyypillisesti hallintaohjelmistot tarjoavat graafisen käyttöliittymän verkkoon ja sen laitteisiin ja ovat konfiguroitavissa eri tavoin.

Verkonhallinnan laajentuessa myös yleiseksi verkossa olevien laitteiden hallinnaksi, ovat hallintaohjelmistot sopeutuneet tilanteeseen hyvin. Esimerkiksi lisättäessä

uudentyyppinen SNMP-valvottava tulostin valvottavaan verkkoon, kerrotaan hallinta-ohjelmistolle tulostimen osoite, ladataan tulostimen valmistajan toimittama MIB-määrittely hallintaohjelmistoon ja asetetaan tarvittavat määrittelyt, kuten hälytys paperin tai musteen loppumisesta ja tulostimen tukkeutumisesta.

SNMP:n rajoitukset

UDP-pohjaisena pollaavana protokollana SNMP kärsii tietyistä rajoituksista. Koska jokainen tietoalkio haetaan agentista erikseen, suurten datamäärien SNMP:llä on hidasta. Sama ongelma vaikeuttaa laajojen tietojärjestelmien hallintaa. Jonkin verran voidaan alkiokohtaisen pollauksen viiveitä kompensoida edistyneemmällä ohjelmoinnilla hallinta-asemassa ja lähettämällä useampia kyselyitä kerralla, mutta esimerkiksi SNMP:n dynaamisten taulukoiden läpikäyntiin tämä ei sovi; kyseiset taulukot muodostetaan dynaamisten MIB-osoitteiden avulla ja ainoa tapa hakea niistä tietoa on GetNextRequest. [Stallings96]

SNMP-agentin lähettämät trapit eivät ole luotettavia sikäli, että agentti ei tarkista niiden perillemenoa eikä UDP-protokollakaan takaa viestin perillemenoa. Täten agentilla ei ole varmuutta hälytyksen perillemenosta.

SNMP-protokollan looginen rakenne ei tue laitteen aktiivista ohjausta. Vaikka ohjattavan laitteen parametrejä voidaan asettaa, SNMP-protokolla ei tue suoraan ohjauskäskyn ja sen parametrien välittämisen käsitettä.

SNMP-mallissa ei ole minkäänlaista hallinta-asemien välistä kommunikaatiota. Hallinta-asemat voivat tietenkin toimia myös agentteina ja siten valvoa toisiaan. Hallinta-asemien välisestä kommunikaatiosta voisi olla kuitenkin hyötyä esimerkiksi hajautettua valvontajärjestelmää rakennettaessa, jolloin valvovat laitteet voisivat keskenään siirtää yhteenvetotietoja valvomistaan osa-alueista.

SNMP ei ota kantaa tietoturvaan eikä reaaliaikaisuuteen. UDP/IP-protokolla ei myöskään tarjoa reaaliaikaisuutta. Trappeja lukuunottamatta SNMP toteuttaa kuitenkin

varsin vahvaa master-slave -mallia, joka ei täysin sulje pois reaaliaikaisuuden mahdollisuutta.

2.2.4. Muut tietoliikenne- ja etäkäyttövaihtoehdot

Automaation etäohjaustarpeet ratkaistaan tyypillisesti muilla keinoilla kuin verkonvalvontaprotokollien avulla. Tässä esitellään muutama yleisin käytössä oleva vaihtoehto.

Ethernet

Ethernet-lähiverkko on yleisimpiä käytössä olevia lähiverkkoprotokollia. Ethernet tarjoaa IP-protokollalle sen tarvitseman paketinvälitykseen.

Ethernet on kilpavaraukseen perustuva lähiverkkoprotokolla. Ethernet-verkon fyysinen koko on rajoitettu tiettyyn, käytettävästä kaapeloinnista riippuvaan maksimipituuteen. Kilpavarausominaisuudesta johtuen Ethernet-verkkoa ei voi käyttää tiukkaa reaaliaikaisuutta vaativaan tiedonsiirtoon. Ethernet-verkko on myös herkkä verkossa olevien koneiden virhetoiminnoille. Esim. ns. broadcast-myrsky, jossa yksi kone lähettää kaikille verkon muille koneille viestejä, pystyy tukkimaan koko verkon estäen kaiken muun liikenteen.

Vasteajan epädeterministisyys on hallittavissa suljetussa Ethernet-verkossa master-slave -protokollalla (kuten SNMP) ja varmistamalla että verkossa ei ole muuta liikennettä. Kun kaikki liikenne on master-slave pohjaista ja verkossa on vain yksi masterkone, ei törmäyksiä voi syntyä. Käytännössä voidaan myös saavuttaa tilastollinen varmuus noin sekunnin luokkaa olevista vasteajoista pitämällä verkon kuormitus alle 10% sen kapasiteetista. [comp.realtime]

Ethernetin fyysiset rajoitukset merkitsevät, että sitä ei voida käyttää laajempiin ratkaisuihin, etenkään ei globaalin mittakaavan ratkaisuihin.

Kenttäväylät

Ohjelmoitavat logiikat tarvitsevat toimiakseen järjestelmästä tietoa keräämiä antureita. Perinteisesti eri valmistajien logiikat ovat edellyttäneet eri tyyppisiä antureita, jolloin järjestelmän suunnittelijan on varsin varhaisessa vaiheessa valittava käyttämänsä tuoteperhe.

Kenttäväylä on yleisnimi automaatioteollisuudessa käyttöön tulossa oleville anturoinnin ja prosessinohjauksen tarpeisiin suunnitelluille tietoliikenneväylille. Kenttäväylästandardeja on useita, mutta tyypillistä useimmille niistä on [Pyyskänen95]:

- valmistajariippumattomuus
- millisekuntien suuruusluokkaa olevat vasteet
- taattu vasteaika eli reaaliaikaisuus
- rajoitettu viestinpituus, kymmeniä tai satoja tavuja
- rajoitettu osoiteavaruus, verkkosegmentissä tyypillisesti enintään kymmeniä tai satoja noodeja
- tiedon siirtonopeus yleensä enintään 2 Mbps
- ei tietoturvallisuutta tai sovellustason protokollapalveluita
- ei yhteydellisiä protokollia
- token- tai master-arkkitehtuuri vasteiden takaamiseksi
- fyysinen media kuuluu väylämäärittelyyn
- fyysinen määrittely teollisuusolosuhteisiin sopiva (häiriönsieto, liittimet, tehonkulutus)

Kenttäväylien standardointi on edelleenkin käynnissä eikä kilpailevista vaihtoehdoista ole vielä ilmaantunut johtavaa standardia.

WWW (World Wide Web)

Internet-verkon parhaiten tunnettu ja näyttävin palvelu on World Wide Web -hypertekstijärjestelmä. WWW:n hypertekstidokumentteihin voi sijoittaa tekstiä, kuvia, linkkejä muihin dokumentteihin samassa tai toisissa palvelimissa ja muita multime-

diaominaisuuksia.

WWW-järjestelmässä dokumentin rakenne kuvataan HTML-kielellä (Hypertext Markup Language), joka on SGML:n (Standar Generalized Markup Language) alijoukko. Tyypillisiä HTML-kielen rakenne-elementtejä ovat eri tasoiset väliotsikot tai tekstin joukossa olevan listan elementit. WWW-järjestelmää käytettäessä käyttäjän selainohjelma hakee HTML-dokumentin palvelimelta käyttäen HTTP-protokollaa (HyperText Transfer Protocol) ja muotoilee dokumentin ulkoasun käyttäjän katselulaitteelle sopivaksi. HTTP on yksinkertainen TCP:n päälle rakennettu tiedostojensiirtoprotokolla.

WWW ei ole kovinkaan sopiva protokolla sulautettujen järjestelmien etähallintaan, koska WWW-järjestelmässä ei ole juuri mitään mikä tukisi valvottavan ja valvovan sovelluksen välistä tiedonsiirtoa.

HTTP-protokolla siirtää tiedostoja. HTML-kielellä määritellään hypertekestitiedoston rakenne. Etähallintaan tarvitaan tietoa valvottavassa laitteessa olevista muuttujista, niiden arvoista, raja-arvoista jne., mielellään yleiskäyttöisessä standardiformaatissa, jotta sama hallintasovellus voisi valvoa eri laitteita ja jopa eri tyyppisiä laitteita. WWW ei tarjoa tätä.

Luonnollisesti HTTP-protokollalla voidaan siirtää parametritiedostoja ja HTML-kieli mahdollistaa parametrien koodauksen, mutta tämä ei riitä; hyötykäyttö hallintaan vaatii lisää standardeja.

Etähallintajärjestelmän käyttöliittymäksi WWW on sen sijaan erittäin sopiva. Laitteistoriippumattomana standardina se sopii hyvin hallintainformaation esittämiseen graafisessa tai tekstuaalisessa muodossa.

Tuotekohtaiset etäkäyttöyhteydet

Automaatiolaitteiden valmistajat ovat tarjonneet jo pitkään erilaisia etäkäyttömahdollisuuksia laitteilleen, tyypillisesti kuitenkin nämä perustuvat valmistajien omiin suljettuihin ja jopa salaisiin standardeihin. Etäkäyttö onkin tarkoittanut usein valmistajan

omaa protokollaa käyttävän RS-232 -sarjayhteyden jatkamista modeemilla toiseen toimipisteeseen. Mobiilikäyttö on saavutettu käyttämällä lankaverkon puhelimen sijaan matkapuhelinta. Nämä käytännössä suljetut ratkaisut eivät ole kovinkaan kiinnostavia tämän tutkimuksen kannalta.

2.2.5. Yhteenveto tietoliikenneteknologioista

Kuten kenttäväylien ominaisuuksista nähdään, kenttäväyläkonsepti on optimoitu tarkoitukseensa automaatioalalla ja prosessinohjauksessa. Erityisesti taatut, lyhyet vasteajat ovat oleellisia reaaliaikaisten prosessien ohjauksessa. Internet- ja Ethernet-protokollat eroavat juuri tässä kenttäväylistä. Korkeamman tason protokollana Internet ei ota mitään kantaa vasteaikoihin. Törmäyksen tunnistukseen ja kilpavaraukseen perustuvassa Ethernet-väylässä ei myöskään voida taata vasteaikoja, joskin suljetussa verkossa voidaan varsin pitkälle olettaa verkon tarjoavan tietyt vasteajat. Kiintoisa ratkaisu reaaliaikaiseen valvontaan ja ohjaukseen voisi olla SNMP-protokollan käyttö suoraan kenttäväylässä.

3. Tarpeet ja sovellukset esitetylle ratkaisulle

Tämän työn hypoteesi on Internetin piirissä kehitettyjen työkalujen, käyttöjärjestelmien ja standardien käytön laajentaminen fyysisten järjestelmien ohjaukseen, kuten tuotantoautomaatioon, logistiikkaan tai kiinteistöhallintaan. Tässä luvussa tarkastellaan muutamaa näistä sovellusalueista.

3.1. Teollisuusautomaation etävalvonta

Tässä diplomityössä toteutetussa virvoitusjuoma-automaatissa ideana on juuri laitteen tilan valvonta. Automaattia ei tarvitse käydä täyttämässä määräajoin, vaan keskusvalvonta voi seurata automaatin tilaa ja lähettää täydennyksen vasta tarvittaessa. Koska valvonta suoritetaan käyttäen julkista tietoverkkoa, on automaatin tila myös sen käyttäjien luettavissa, jolloin automaatin käyttäjä voi varmistua haluamansa merkkisen tuotteen saatavuudesta ja lämpötilasta. Lisäksi on työssä toteutettu automaatin etäohjaus. Käyttäjä tilaa juoman verkon kautta eikä itse automaatissa ole normaalia käsikäyttöistä ohjauspanelia laisinkaan.

Edempänä esitettyjen logististen näkökulmien lisäksi automaatti on myös esimerkki fyysisestä laitteesta, jonka tilaa ja toimintavalmiutta etävalvotaan. Esimerkiksi konepajan automaattisorvit voisivat olla SNMP-valvottuja. SNMP-agentti valvoo anturien avulla sorvin tilaa eri tavoin, esimerkiksi mittaamalla jonkin keskeisen laakerin lämpötilaa. Hallinta-asema hakee tiedot säännöllisin väliajoin ja huomatessaan laakerin käyvän tavallista lämpimämpänä, se lähettää ilmoituksen huolto-organisaatiolle.

Mikäli laakerin lämpötila ylittää sallitut raja-arvot, sorvin agentti voi trap-toiminnon kautta lähettää tiedon suoraan hallinta-asemalle (odottamatta hallinta-aseman normaalia pollausta) ja omatoimisesti pysäyttää sorvin. Hallinta-asema voi lähettää modeemilla viestin huoltomiehen hakulaitteeseen ja tulostaa vikaraportin tehdassalin kirjoittimelle.

Tämä ei sinänsä ole mitenkään mullistavaa. Vastaavia toimintoja on toteutettu perin-

teisilläkin ohjelmistoilla. Oleellista tässä yhteydessä on standardiprotokollien käyttö. SNMP on yleistyvä protokolla ja sen avulla voidaan tulevaisuudessa valvoa organisaation kaikkia verkossa olevia laitteita. Tällöin sama verkonvalvonta-asema voi valvoa organisaation infrastruktuuria sen eri tasoilla, tietoliikennekomponenteista paperikoneisiin.

Valvonta-asemia voi myös olla useampia ja ne voivat toimia organisaation eri tasoilla. Esimerkiksi tuotannon kunnossapito seuraa tuotantolaitteiston päivittäistä tilaa, kun taas yrityksen hallinto kerää kattavaa tilastoa laitteiston käyttöasteesta.

Etävalvonta voidaan erottaa varsinaisesta ohjauksesta ja tietoa voidaan välittää organisaation ulkopuolelle. Esimerkiksi paperikonevalmistajat ovat kiinnostuneet seuraamaan asiakkaille toimitettujen koneiden kuntoa ja asetusarvoja, vaikka kone onkin asiakkaan hallussa. [Nyberg95]

SNMP-protokollalla on rajoituksensa eikä se sovellu kaikkeen, mutta malli, jossa automaation varsinainen ohjaus toteutetaan kenttäväylillä, suurivolyymisen ohjausdatan (ohjelmat, mallit) siirto FTP:llä tai muulla sopivalla protokollalla ja laitteen tilan seuranta ja ohjausparametrien muutos SNMP:llä, vaikuttaa kiinnostavalta ja mahdolliselta.

3.2. Kiinteistöhallinta

Kiinteistöautomaatio on 1980-luvun alusta kehittynyt yhä merkittävämmäksi osaksi kiinteistöjen hallintaa. Kiinteistöautomaation avulla voidaan saavuttaa huomattavia kustannussäästöjä hallitsemalla rakennuksen energiankulutusta tehokkaasti, VTT:n arvioinnin mukaan jopa 5 kWh rakennuskuutiometriä kohti vuodessa [Pyyskänen95]. Liike- ja toimistorakennuksissa muuntojoustavuus on toinen tärkeä kustannussäästöjä tuottava tekijä.

Normaaliin kiinteistöhuoltoon kuuluu myös kiinteistön hallintajärjestelmän valvonta. Kiinteistöhuollon siirtyessä yhä enemmän kiinteistöhuoltoyrityksille, on valvonta

mielekästä suorittaa etävalvontana. Harvassa kiinteistössä on muutenkaan omaa henkilökuntaa paikalla 24 tuntia vuorokaudessa.

Kiinteistön etävalvonta on tarpeellista, paitsi vikatilanteiden takia, myös päätöksentekoon tarvittavan tiedon keruuta varten. Kiinteistöjohtamisen kehittyessä yhä kattavammaksi, tarvitaan sen tueksi analyysejä ja evaluaatioita, joihin puolestaan tarvitaan kattavaa tietoa kiinteistöstä ja sen käytöstä. Tämä merkitsee myös sitä, että eri kiinteistöjä edustavan datan tulee olla keskenään vertailukelpoista.

1990-luvun puolivälin paikkeilla kiinteistöjen etävalvontajärjestelmät olivat vielä valmistajakohtaisia suljettuja järjestelmiä, mutta vuosikymmenen loppupuolella on kehityksen suuntana siirtyminen avoimiin standardeihin [Paavilainen97]. Hallitsevaan asemaan näyttää Suomessa nousevan LonWorks-niminen kenttäväyläteknologia.

Kenttäväyläpohjainen ratkaisu on hyvä anturointiin ja kiinteistön sisäiseen tietoliikenteeseen, mutta koska kiinteistöt ovat pääsääntöisesti erilaisia, on niiden anturointikin erilaista.

LON-malli ei tarjoa korkeamman tason abstraktiokerrosta, vaan jokainen kiinteistö on kuvattava erikseen valvonta-asemassa. [Sahlstén99] Samoin korkeamman tason tilastotiedon kerääminen hankaloituu, koska kunkin kiinteistön anturointitieto on muutettava vertailukelpoiseen muotoon.

3.3. Logistinen näkökulma sulautettuihin järjestelmiin

Logistiikka on oppi tavara- ja tietovirtojen käsittelystä. Perinteisesti logistiikka on keskittynyt kuljetukseen ja varastointiin. Moderni logistiikka pyrkii optimoimaan kuljetuksia ja minimoimaan varastointia, mm. hyödyntäen organisaatioiden rajat ylittäviä tietojärjestelmiä.

Logistiikan kustannukset ovat suuret. Suomessa logistiikan kokonaiskustannukset yrityksille vaihtelevat 5-20% liikevaihdosta. Varastointikustannuksien laskeminen on vaikeaa, mutta yleensä arvioidaan varastoinnin vuosikustannuksiksi 20-50% varaston

arvosta. On siis selkeää, että logistiikkaa tehostamalla voidaan saavuttaa merkittäviä säästöjä. [Sakki94]

Perinteinen välivarastoihin ja tilausrajoihin perustuva logistiikkaketju aiheuttaa tuotantoon voimakasta aaltoilua (Burbidge-effekti). Mikäli tuotannonsuunnittelu on tietoinen kulutuksesta, se voi ennakoida väliportailta aikanaan tulevat tilaukset tai jopa omatoimisesti toimittaa tuotteen asiakkaalle.

Moderni logistiikka ja tuotannonohjaus perustuu imuohjaukseen, jossa asiakkaan tarpeet ohjaavat tuotantoa ja logistiikkaketjua. JOT-ohjaus on tyypillinen esimerkki tästä periaatteesta. Asiakas toivoo yleensä mahdollisimman nopeita toimitusaikoja, logistiikkaketju puolestaan pyrkii minimoimaan varastot ja niihin sitoutuneen pääoman. Näiden keskenään jonkin verran ristiriitaisten tavoitteiden saavuttamista voidaan helpottaa nopeuttamalla informaation kulkua asiakkaan tarpeista tuotantoketjun toiseen päähän.

Imuohjauksen toteutus riippuu toimialasta ja logistiikkaketjun rakenteesta, mutta esimerkiksi nykyiset juoksevaa inventaariota toteuttavat vähittäismyyntipäätteet mahdollistavat tuotetilausten automatisoinnin kassapäätteestä saatujen myyntitietojen perusteella. Tuotannonsuunnittelulle on arvokasta saada tietoa kulutuksesta jo ennen varsinaista tilausta, jotta aikanaan tulevat tilaukset voitaisiin ennakoida.

Sulautettujen järjestelmien puolelta esimerkkinä logistiikkaketjun tukemisesta on Raha-automattiyhdistyksen kolikkopelien valvontajärjestelmä, jossa pelit tiedottavat puhelimitse tilastaan. Logistiikkanäkökulmasta pelit tarvitsevat rahastusjärjestelmän tyhjentämistä ja oikean suuruisia kolikoita voitonmaksuun. Lisäksi on otettava huomioon, että toimimaton peli ei tuota.

Prototyyppinä toteutettu virvoitusjuoma-automaatti tarvitsee täyttämistä. Virvoitusjuoman kulutuksen ennustaminen ei ole helppoa, siihen vaikuttavat mm. sää ja ihmisvirrat automaatin ohitse. Etähallinnalla voidaan täytöt ajoittaa oikeaan tarpeeseen, jolloin automaatti pysyy jatkuvasti myyntivalmiudessa ja turhat käynnit minimoidaan.

Lisäksi automaattien etäseurannalla voidaan kerätä reaaliaikaista tietoa kulutuksesta ja eri tuotteiden menekistä, josta on hyötyä sekä tuotannonohjaukselle että markkinoinnille.

4. Prototyypin kehittäminen

4.1. Suunnittelu

Projekti lähti liikkeelle kun Pekka Nikander ilmoitti että Nixu Oy:n limuautomaatile on tehtävä jotain, koska siinä on yli kolme nappulaa eikä IP-osoitetta. Triviaaliratkaisu, jota esitin ensimmäiseksi, olisi ollut Linux-PC, jonka kirjoitinportista otettaisiin ohjaus limuautomaatlle ja johon tehtäisi yksinkertainen WWW-käyttöliittymä C:llä tai Perlillä. Pekka halusi kuitenkin toteuttaa asian teknisesti kunnianhimoisesti.

Mach oli vuoden 1994 syksyllä ns. kuuma käyttöjärjestelmä, Open Software Foundation oli toteuttamassa Unixia sen pohjalta, IBM:n oltiin kuultu olevan kiinnostunut mikrokerneleistä ja tutkimuspuolella järjestettiin Mach-konferensseja. Alustavan kirjallisuustutkimuksen mukaan Mach vaikutti teknisesti kiinnostavalta käyttöjärjestelmältä mikrokerneliarkkitehtuureineen [Tanenbaum92][Boykin93], se oli saatavilla, Teknillisessä korkeakoulussa oli samaan aikaan käynnissä jokunen Machiin liittyvä projekti ja Mach käyttöjärjestelmänä on riittävän lähellä Unixia, mutta silti sopivan erilainen. Kaikkien näiden seikkojen perusteella valittiin Mach ratkaisun toteutuskäyttöjärjestelmäksi. Muita vaihtoehtoja olisivat olleet jokin Unixin versio, mistä ei olisi opittu paljoakaan uutta tai kaupallinen QNX, joka olisi ollut kallis ja sidottu i386-arkkitehtuuriin.

Tietoliikenteen suhteen Internet oli jo vuonna 1994 selvä valinta. Vaikka Internet-boomi oli vasta alussaan, uskomme Internetiin tulevaisuuden kattavana tietoverkkona oli jo tuolloin vahva. Internet-protokollien suhteen oli tarjolla kolme mielekästä vaihtoehtoa: käyttää WWW:tä, verkonhallintaprotokollia tai tehdä oma protokolla. Kuten aiemmin on todettu, WWW ei skaalaannu. Oman protokollan kehittäminen ei ole mielekästä, jos voi käyttää jotain valmiiksi mietittyä ja standardoitua. Verkonhallintaprotokollat vaikuttivat käyttökelpoisilta, reitittimen ja virvoitusjuoma-automaatin välillä kun ei ole suurtakan eroa.

Verkon hallintaan on olemassa SNMPv1, SNMPv2 ja CMIP-protokollat. CMIP julis-

tettiin nopeasti liian raskaaksi ja vaikeaksi toteuttaa tähän työhön. SNMPv2 oli liian hajanainen ja keskeneräinen standardi käytettäväksi, myöhemmin se jakautuikin kolmeksi versioksi. Jäljelle jäi siis toimiva, joskin rajoittunut SNMPv1.

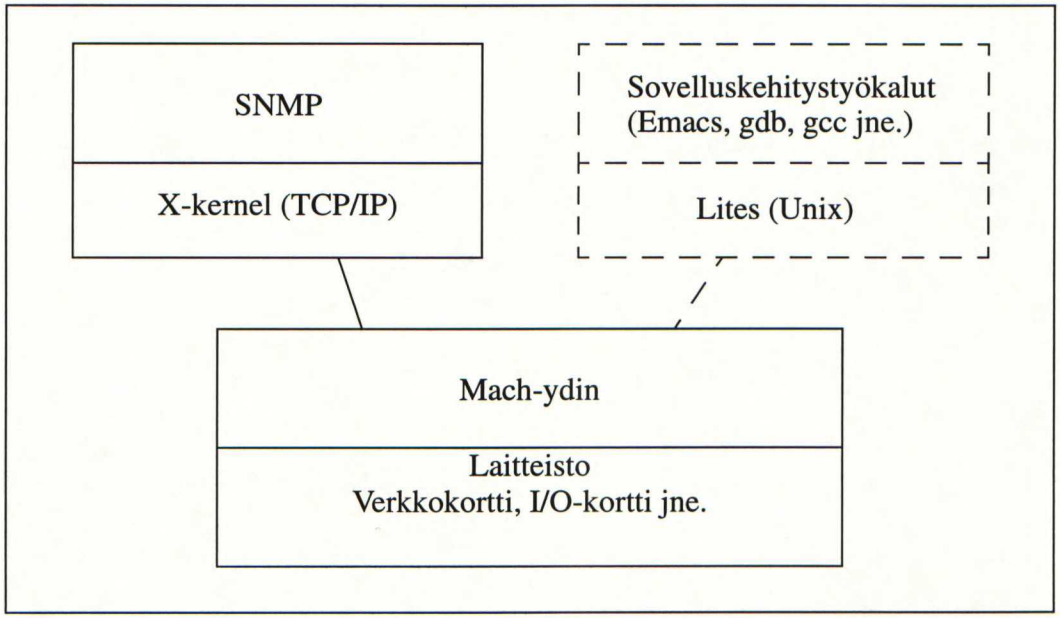
Laitteistoksi olisi voinut pohtia jotain muutakin ympäristöä kuin PC:tä (i386), mutta käytännössä PC-laitteistoympäristöä pidettiin yksinkertaisimpana valintana. PC-arkkitehtuurin mukaisia komponentteja kun on saatavilla erilaisina: tehokkaita pöytäkoneita ja teollisuus-PC:itä, PC-104-korteista koottavia pienikokoisia ja helposti sijoitettavia laitteita jne. Lisäksi PC-arkkitehtuuriin saa erilaisia lisälaitteita, kuten I/O-kortteja, Flash-ROM-muisteja jne. Massatuotetut PC-laitteistot ovat myös edullisia.

Alustavan suunnittelun jälkeen pohdittiin vielä tarkemmin haluttua arkkitehtuuria. Tavoitteena oli yhdistää kehitysympäristö ja varsinainen toteutusympäristö mahdollisimman pitkälle. Kun alla on yleiskäyttöinen mikrokernelikäyttöjärjestelmä (mahdollisin reaaliaikaominaisuuksin) voidaan sen päällä ajaa Unix-emulaattoria ja täten käyttää kaikkia Unixille saatavilla olevia sovelluskehitystyökaluja. Kun järjestelmä saadaan valmiiksi, voidaan Unix-emulaattori jättää pois ja kehitetty sulautettu sovellus jää itsenäiseksi Mach-tehtäväksi mikrokernelin päälle.

Machin valintaan oli vaikuttanut myös se, että Johannes Helander oli juuri julkaissut omana diplomityönään Machille tekemänsä Lites-nimisen Unix-emulaattorin, jonka avulla saadaan Unixin toiminnot Mach-ympäristöön. Olemassa olevat SNMP-toteutukset, joista jotakin suunniteltiin käytettäväksi, edellyttävät allaan olevan TCP/IP-toteutuksen ja Socket-rajapinnan. Tämä suunniteltiin toteutettavan Arizonan yliopiston X-kernel -tietoliikennepaketilla, joka siirrettäisiin Mach-ympäristöön.

Projektin alussa, talvella 1995 suunniteltu järjestelmä vaikutti allaolevan kaltaiselta,

Lites ja sovelluskehitystyökalut ovat poistettavissa kun järjestelmä on valmis:



Kuva 6. Suunniteltu järjestelmän rakenne.

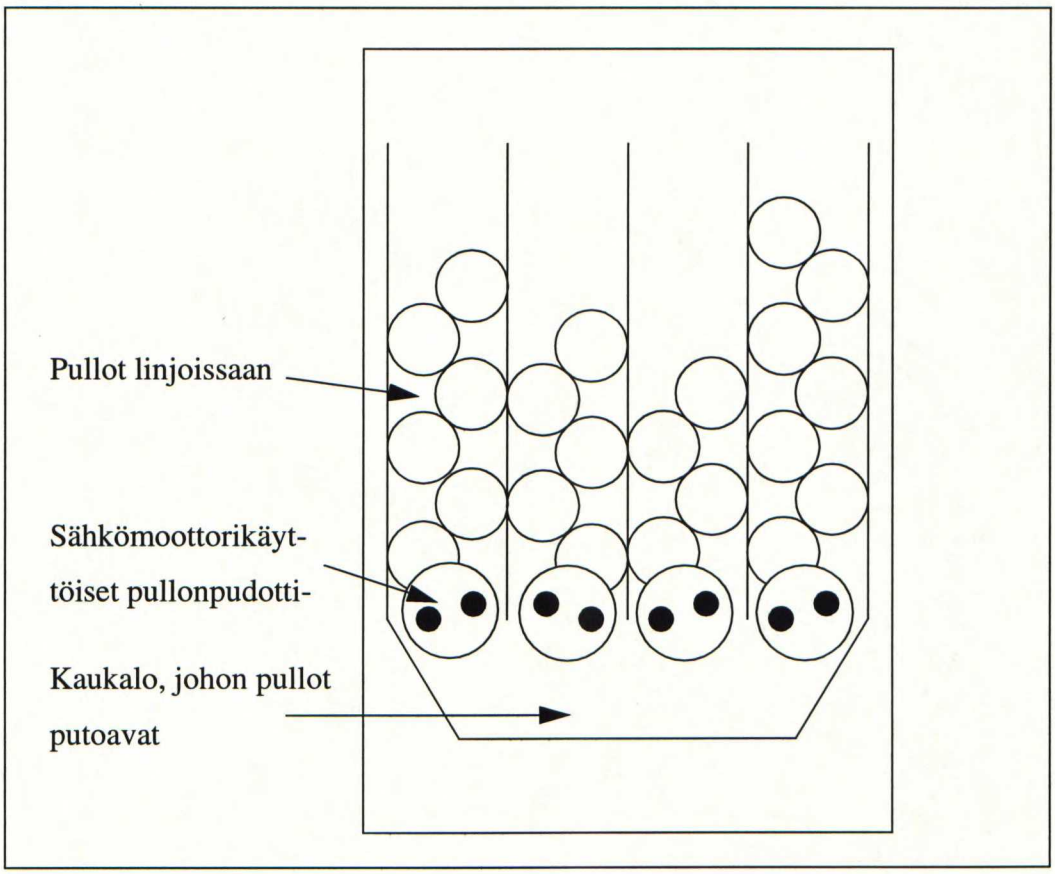
4.2. Prototyyppilaitteisto

Kehityslaitteistona käytettiin tavallista toimisto-PC:tä, jossa oli 100 MHz i486 keskusyksikkö, 32 Mt muistia, 2 kappaletta 700 Mt:n kovalevyä, geneerinen näytönohjain ja ACL-724-tyyppimerkintäinen digitaalinen I/O-kortti, josta saadaan 8 optisesti erotettua (optoisoloitua) relelähtöä ja 16 tuloa. Koska Machin laiteohjaimien valikoma on suppea, oli verkkokortiksi etsittävä vanha Western Digital 8003.

Lopullinen tuotantokäytössä oleva ohjausyksikkö on myöskin toimisto-PC, jossa on 33 Mhz i386-keskusyksikkö, 8 Mt mustia ja 700 Mt:n kovalevy. Lopulliseen toteutukseen olisi kelvannut vähätehoisempikin kone, mutta tehottomampaa ei Nixu Oy:n varastosta enää löytynyt.

Käytetty virvoitusjuoma-automaatti koostuu neljästä juomalinjasta, joihin pullot sijoi-

tetaan.



Kuva 7. Virvoitusjuoma-automaatin mekaaninen rakenne

Sähköisesti virvoitusjuoma-automaatti on toteutettu 220 V vaihtovirtaa käyttävällä relelogiikalla [Pekurinen94]. Kolikon hyväksyttyään rahastusyksikkö lähettää signaalin releelle, joka yhdistää etupanelissa olevat neljä juomanvalintanappia pullonpudottimien moottoreihin. Valintanapin painaminen käynnistää pullonpudottimen moottorin sekä vapauttaa releen, joka yhdisti valintanapit moottoreihin. Pullonpudotin pyörähtää puoli kierrosta pudottaen pullon ja pysähtyy sen jälkeen.

Juoma-automaatin ohjaamiseen tarvitaan viisi relettä. Yksi, joka emuloi kolikontunnistimen hyväksymissignaalia ja neljä juomanvalitsinnappeja emuloivaa relettä. Tarkkoja ohjausvaatimuksia pullon pudottamiseksi ei tutkittu, pikainen kokeilu osoitti että 0,1 sekunnin oikosulkusignaali kolikontunnistimelta, 0,1 s tauko ja 0,1 signaali juomanvalitsinpainikkeelta toimii. Projektin aikana suoritetussa tutkimuksessa todettiin,

että 0,3 sekunnin operointiaika tarjoaa riittävän reaaliaikaisen vasteen kohderyhmänä toimineelle Nixu Oy:n henkilöstölle, joka testasi järjestelmän toimintaa useiden tuhansien pullojen verran.

4.3. NetBSD

Kehityslaitteiston asennus alkoi NetBSD-Unixin asennuksella. NetBSD on yksi useista vapaasti saatavilla olevista Unix-toteutuksista. Muita tunnettuja vastaavanlaisia ovat mm. Linux ja FreeBSD. NetBSD valittiin, koska sitä oli käytetty Litesin kehityksessä ja koska se vaikutti varsin stabiililta tuohon aikaan. Esim. nykyään suosittu Linux ei ollut yhtä kehittynyt vielä tuolloin.

NetBSD:n asennus oli varsin suoraviivaista. Verkosta haettiin kaksi tiedostoa, jotka olivat suoria kuvia asennuslevykkeistä. Tiedostot kopioitiin levykkeille ja PC käynnistettiin levykkeeltä. Levykkeeltä käynnistyi minimaalinen NetBSD-ympäristö, jossa yksinkertainen shell-komentotiedosto ohjasi käyttäjän asennuksen tärkeimpien vaiheiden läpi, kuten levyn alustus, tietoliikenneyhteyksien konfigurointi, NetBSD:n ohjelmistojen hakeminen FTP:llä verkkopalvelimelta jne. Tuloksena oli kohtuullisen karu Unix-ympäristö, jossa on kaikki perusohjelmistot.

Seuraavaksi hain ftp.funet.fi-palvelimelta Emacs-editorin, GNU-utiliteetit (`make` jne.) ja gcc- C-kääntäjän Machin kääntämiseen sopivan version. Tämän jälkeen koneessa oli käyttökelpoinen Unix-ympäristö ja valmius Machin asentamiseen.

4.4. Mach

Kirjallisuustutkimus loi Mach-mikrokernelistä kokemattomalle lukijalle ihannoivan kuvan. Kirjallisuuden mukan Mach-käyttöjärjestelmässä on seuraavat ominaisuudet:

- pienikokoinen mikrokerneli
  - todellisuudessa ydin on kooltaan noin yksi megatavu, mikro viittaa ytimen toimintoihin, ei kokoon.

- tuki eri käyttöjärjestelmille
  - Mach 3:n Unix-tuki edellyttää AT&T:n Unix-lisenssiä.
  - MS-DOS-tuki on todellakin vain MS-DOS-rajapinta, ei PC-rajapinta. Ns. MS-DOS-ohjelmista suurin osa olettaa käytettävissä olevan kaikki PC-laitteiston ominaisuudet.

Tässä kohdassa opin ensimmäisen merkittävän asian. Tieteellisiä julkaisuja pitää lukea hyvin huolellisesti. Kun jotakin on onnistuttu tekemään jossain, se ei tarkoita, että saavutettu tulos olisi kovinkaan yleiskäyttöinen, ylläpidettävä tai yleisesti saatavilla.

Todellisuudessa Mach-käyttöjärjestelmää on kehitetty eri korkeakouluissa eri suuntiin, välillä painottuen uuteen Unixiin, välillä mikrokerneliarkkitehtuurin ominaisuuksiin.

4.4.1. Mach4 UK02p21:n asennus

Kun NetBSD oli asennettu haettiin osoitteesta ftp://flux.cs.utah.edu/flux/mach/ Machin lähdekoodi ja tarvittavat käännöstyökalut.

Ristiinkääntäjällä käännettiin ensin Machin ydin ja kirjastot. Mach4:n lähdekoodi on jaettu kahteen hakemistopuuhun, laitteistoriippuvaan (esim i386) ja laitteistoriippumattomaan. Käännös suoritetaan kolmanteen hakemistopuuhun, ensin alustaen käännös GNU-Autoconfigure-ohjelmalla ja sitten Unixin normaalilla `make`-ohjelmalla.

Käännös perustuu ensisijaisesti laitteistoriippuvaan koodiin ja puuttuvat osat otetaan laitteistoriippumattomasta. Asennusdokumentin mukaan käänteinen järjestys osoittautui hankalaksi, koska laitteistoriippumattoman koodin on hankala tietää, milloin tarvitaan laitteistoriippuvia osia.

Mach kääntyi kohtuullisen jouhevasti, mutta onnistumisen testaaminen jäi odottamaan myös Litesin kääntämistä.

4.5. Lites

Kirjallisuudessa esitetty [Boykin93] Unix Machin päälle edellyttää käytännössä AT&T:n lähdekoodilisenssiä. Lites on Johannes Helanderin Teknillisessä korkeakoulussa diplomityönään kirjoittama vapaasti jaettava Unix-toteutus Mach-käyttöjärjestelmän päälle. Lites tarjoaa Unix-ytimen palvelut monisäikeisessä palvelinprosessissa, joka on Machin näkökulmasta yksi tehtävä. Lites ei ole täydellinen Unix, vaan Lites-ydin käyttää muiden Unixien (NetBSD, FreeBSD, 386BSD, Linux) binäärejä, ollen rajapinnaltaan yhteensopiva näiden käyttöjärjestelmien kanssa. Koska kehitys-laitteistoon oli jo asennettu NetBSD, Lites pystyi käyttämän sen binäärejä ja asetuksia.

Litesin asennuksen jälkeen kone pystytään käynnistämään joko NetBSD:n ytimellä tai Machin ytimellä, joka käynnistää Litesin. Sekä NetBSD että Mach-Lites yhdistelmä käyttävät samaa tiedostojärjestelmää ja samoja binäärejä (poislukien ytimen tietorakenteisiin sidotut ohjelmat, kuten `ps`). Tavallinen käyttäjä ei välttämättä huomaa onko käytössä Lites-ydin vai NetBSd-ydin.

Koska Lites-palvelin ei asioi suoraan laitteiston kanssa, vaan tekee sen Mach-ytimen kautta, voidaan samassa tietokoneessa ajaa useampia samanaikaisia Lites-palvelimia. Tämä tarjoaa uusia näkemyksiä luotettavuuteen ja tietoturvallisuuteen. Vahinko voidaan rajata erittäin tehokkaasti yhteen virtuaaliympäristöön, mikäli itse Machin suojaukset ovat riittävät.

Lites koostuu päätasolla kahdesta ohjelmistokomponentista, palvelimesta ja emulaattorista. Palvelin tarjoaa Unixin tavalliset palvelut (tiedostojärjestelmä, tietoliikenne jne). Emulaattori luetaan ohjelmaa ladattaessa jokaisen suoritettavan ohjelman koodiin mukaan, dynaamisten kirjastojen tapaan, ja emulaattori kääntää ajettavan ohjelman systeemikutsut palvelupyynnöiksi Lites-palvelimelle.

Litesin kehitys näyttää päättyneen vuonna 1996 versioon lites-1.1.u3

4.5.1. Litesin asennus

Johtuen omasta vähäisestä kokemuksestani käyttöjärjestelmien parissa ja Machin vähäisestä dokumentoinnista, käyttöjärjestelmän pystyttäminen osoittautui melko työlääksi. Verrattuna esimerkiksi tyypilliseen Linux-asennuspakettiin, Mach+Lites-ympäristön pystyttäminen on arviolta kertaluokkaa vaativampi tehtävä. Törmäsin mm. sellaisiin ongelmiin kuin, että NetBSD:n versioiden 1.0 ja 1.0A binääriformaatti on erilainen ja vaikka minulla oli tiedossa tarvittavat muutokset Litesin koodiin, jotta Lites osaisi hyödyntää uutta binääriformaattia, meni viikko aikaa ennen kuin ymmärsin mistä on kyse. Asiaa selvitellessäni en mm. ymmärtänyt oliko ongelma Litesissä vai Machissa, koska en tiennyt miten Mach-ydintä voitaisiin testata sellaisenaan.

Lopulta asennus onnistui ja käytettävissä oli Lites-ympäristö ja sen alla oleva Mach.

4.5.2. Lites ja Mach yhteistoiminta

Kun koneessa on kaksi käyttöjärjestelmää (Mach ja Lites) ja kun Lites-ympäristössä kehitetään ohjelmistoja, jotka toimivat suoraan Machin päällä, olisi luonnollisestikin mukavaa pystyä luomaan ohjelma, joka kommunikoisi molempien ympäristöjen kanssa. Saatavilla oleva UX-binaries -paketti sisältääkin juuri tällaisia ohjelmia. Kuitenkin huolimatta useiden päivien yrityksistä en onnistunut itse tekemään ohjelmaa, joka toimisi samanaikaisesti molemmissa ympäristöissä. Ongelma liittyy Machin ja Litesin kirjastoihin ja niiden linkitykseen käännettyyn ohjelmaan. Toimiakseen molemmissa ympäristöissä, olisi ohjelmaan linkitettävä Litesin normaalikirjastot, kuten I/O-toiminnot ja lisäksi Machin porttirajapintaa tukevat toiminnot. Tämä ei kuitenkaan onnistunut, vaan törmäsin aina keskenään ristiriitaisiin systeemikutsuihin. Jouduin tyytymään siihen, että voin käynnistää Lites-ympäristöstä Mach-ohjelman, jonka kanssa voin keskustella koneen konsolilta (PC:n näyttö ja näppäimistö), mutta jonka tulostusta en saa siirrettyä Lites-ympäristöön.

Epäilemättä ongelma olisi ollut ratkaistavissa riittävällä osaamisella, mutta tässä kohdassa oma osaamiseni loppui kesken enkä pitänyt ongelmaa riittävän suurena lähteäk-

seni häiritsemään muita sillä.

Mach+Litesin ja NetBSD:n (tai jonkin muun Unixin) yhteiselossa on tiettyjä käytännön ongelmia. Lites ei tarjoa täysin samaa rajapintaa kuin normaali Unixin ydin, esimerkiksi virtuaalimuistista ja prosesseista ei ole saatavilla samoja tietoja.

4.5.3. Laiteohjain Machiin

Työhön käytetty Adclone ACL-724 digitaalinen I/O-kortti on x86-ympäristössä erittäin helppo käsiteltävä ohjelmoijan näkökulmasta. Kortin tulot ja lähdöt sijoittuvat prosessorin erilliseen I/O-osoiteavaruuteen luettaviksi ja kirjoitettaviksi tavuiksi. Lisäksi on yksi tavu, jolla kortti konfiguroidaan. Kortissa on 24 digitaalista I/O-porttia, jotka ovat kahdeksan ryhmissä jaettavissa tuloiksi tai lähdöiksi, riippuen käytetystä terminaalikortista. Käyttämässämme terminaalissa on 8 optoisoloitua relelähtöä ja 16 tuloa. Käytännössä laiteohjaimen on siis pystyttävä kirjoittamaan yksi tavu käynnistyksen yhteydessä ja sen jälkeen lukemaan kolmea tavua ja kirjoittamaan yhtä.

Jotta I/O-korttia käytettäisiin oikeaoppisesti Machin systeemikutsurajapinnan kautta, on sille kirjoitettava laiteohjain. Laiteohjaimen pohjaksi valittiin sarjaliikenneportin ohjaimen koodi. Koska I/O-kortin kautta ei kulje tietovuota, käytetään kortin tilan lukemiseen ja muuttamiseen laiteohjaimen statusta käsitteleviä systeemikutsuja. Laiteohjaimen kohdalla mentiin selkeästi yli siitä mistä aita on matalin. Selkeä potentiaalinen ongelma tulee jo siitä, että laiteohjaimen tilaa muutettaessa, on asetettava kaikki lähdöt samalla kertaa. Tämä estää käytännössä sen, että useampi ohjelma voisi ohjata samaa laitetta, ainakin ilman ohjelmien välistä kommunikointia. Tämä olisi ratkaistavissa joko jakamalla eri tehtäville oikeuksia eri lähtöihin tai, jos oletamme että vain yksi ohjelma kerrallaan käsittelee kutakin lähtöä, yksinkertaisen bittimaskin avulla, jolloin kunkin lähdön käsittely ei vaikuta muihin lähtöihin.

Uteliaisuuttani toteutin kuitenkin laiteohjaimeen kortin tilaa pollaavan toiminnon, jolloin säie tehtävässä voi jäädä odottamaan jonkin I/O-linjan tilan muuttumista.

4.5.4. Ohjelmankehitys Lites + Mach -ympäristössä

Machin tehtävien (prosessien) välinen kommunikointi perustuu porteiksi kutsuttuihin tietorakenteisiin ja niitä käsitteleviin systeemikutsuihin. Portti on aina jonkin tehtävän (tai ytimen) omistuksessa ja säikeen oikeus hyödyntää portin palveluita perustuu säikeen kuulumiseen tehtävään. Esimerkiksi virvoitusjuoma-automaattia ohjaavan digitaali-I/O -kortin tila saadaan luettua seuraavalla ohjelmakoodilla:

```
unsigned char a=0, b=0, c=0;
dev_status_t status[20];
kern_return_t rc;
mach_msg_type_number_t status_count;
mach_port_t   device_server_port, acl724_port,
privileged_host_port;

privileged_host_port = task_by_pid(-1);
device_server_port = task_by_pid(-2);
rc = device_open(device_server_port, D_READ | D_WRITE, "acl",
  &acl724_port);

status_count = 3;
rc = device_get_status(acl724_port, 0, status, &status_count);
a = (int) status [0];
b = (int) status [1];
c = (int) status [2];
```

Ohjelmakoodissa hankitaan ensin Lites-palvelimelta oikeus käyttää laiteohjaimia privileged_host_port-kutsulla, tämä edellyttää ohjelman suorittamista Litesin pääkäyttäjän oikeuksilla (root-tilassa). Kun tehtävä on saanut privileged-oikeuden, se pyytää Mach-ytimeltä porttia laiteohjaimista vastaavaan palveluun. Laiteohjain on vielä avattava sekä lukemista ja kirjoittamista varten.

Alustuksen jälkeen luetaan laiteohjaimelta I/O-kortin tulojen ja lähtöjen tila kutsumalla device_get_status-rutiinia ja antamalla sille argumentiksi laiteohjaimeen viittaavan portin osoitteen.

Machin päällä olevassa Lites-ympäristössä on käytettävissä normaalit Unix-ohjelmointityökalut, kuten gcc-kääntäjä ja Emacs-editori. GDB-debuggerista on käytettävissä Machia tukeva ja säikeitä ymmärtävä versio.

4.6. X-kernel

X-kernel on Arizonan yliopistossa kehitetty (tietoliikenne)protokollien implementointiympäristö. Protokollat toteuttava ohjelmakoodi voidaan kirjoittaa täysin laitteisto- ja käyttöjärjestelmäriippumattomaan muotoon. X-kernel tukee erityisesti uusien protokollien kehitystä.

X-kernelin kehityksen takana on myös ajatus siitä että keskusyksiköiden suorituskyky kehittyy nopeammin kuin muistin suorituskyky. X-kernel pyrkii tarjoamaan tehokkaan protokollaimplementaation suurta nopeutta vaativiin sovelluksiin minimoimalla keskusyksikön ja muistin välisen tietoliikenteen, mm. siten, että kerran muistiin luettua dataa ei siirretä muistissa, vaan eri protokollakerrokset käsittelevät samaa puskurialuetta.

4.6.1. X-kernelin asennus

Kun Mach+Lites -käännösympäristö oli pystyssä ja toimi, alkoi X-kernelin siirto Mach-ympäristössä suoritettavaksi tehtäväksi. Työtä aloitettaessa X-kernelistä oli olemassa Unix-käyttöjärjestelmässä ajettava versio ja Machin ytimen sisälle upotettava versio. Tässä työssä kuitenkin haluttiin pitää Machin ydin puhtaana, joten X-kerneli siirrettiin Machiin omaksi palvelimekseen.

Koska X-kernel oli jo aiemmin siirretty Mach-ympäristöön, mutta ytimen sisälle, ei sen siirtäminen ytimen päällä toimivaksi palvelimeksi ollut kohtuuttoman vaikeaa. Machin ytimen sisälläkin X-kernel käyttää Machin tavallista porttirajapintaa, joten suurin osa siirtoa oli NetBSD:n include-tiedostojen määrittelyn sovittamista muille Unixin varianteille tehtyyn ohjelmistoon. Machin päällä ajettu Liteshän käytti NetBSD:n binäärejä ja kirjastoja. Virhetulostuksen seuraamista varten ohjelmiston alkuun lisättiin Machin systeemikutsut, joilla saatiin tulostusoikeus konsolipäätteelle (PC:n näyttö) ja ohjattiin virhetulostus näytölle. Myös kellonaika jouduttiin muuttamaan Machin systeemikutsulla `host_get_time()` haettavaksi Unixin tavallisen `gettimeofday()`-funktion sijaan.

Tässä kohdassa törmättiin myös pieniin ongelmiin, mm. X-kernelin jakeluversiossa eri TCP/IP-protokollien tunnusnumerot olivat väärät. Normaalisti Ethernet-paketin sisällön tyyppi 0x800 tarkoittaa IP-pakettia ja IP-paketin sisällön tyyppi 17 tarkoittaa UDP-pakettia. Koska X-kernel on tarkoitettu ensisijaisesti protokolliin liittyvään tutkimukseen ja kehitykseen, on usein hyödyllistä käyttää tarkoituksellisesti vääriä sisällöntyyppinumeroita tässä yhteydessä, jolloin testattava protokolla ei häiritse varsinaiseen tietoliikenteeseen tarkoitettujen protokollien käyttöä. Jakeluversiossa olisi kuitenkin pitänyt olla oikeat protokollanumerot. Kiinnostavaa on, että ilmeisesti virheellinen ohjelmistopaketti oli ollut jakelussa varsin pitkään, ilman että kukaan olisi huomauttanut virheestä projektin väelle. Oletan tämän johtuvan X-kernelin vähäisestä jakeluvolyymistä. Kyseessähän on tutkimuskäyttöön suunnattu ohjelmisto.

Suorittamani X-kernelin siirto Machiin tehtiin varsin karkeasti, jakelin omaa versiotani muutaman kappaleen aiheesta kiinnostuneille. Työni laatua kuvaa Stephen Clawsonin kommentti mach4-announce-listalla:

```
* Patches to version 3.2 of the x-kernel:

This is a patch to v3.2 of the x-kernel from The University of Ari-
zona.  It includes modifications to the mach3 out-of-kernel code so
that it will compile and run on a Mach4/Lites system, along with some
information about how to configure and build an x-kernel.

Originally these patches were going to be based on the x-kernel pat-
ches posted to mach4-users by Timo Kiravuo (kiravuo@nixu.fi). Howe-
ver, the changes he had to make to build the x-kernel without a libc
(since the patches were designed to build with the i386-mach cross-
build tools) would take much longer to clean up than to start over
and get the x-kernel to build with the *BSD `native' mach4 build
tools, which have a libc that they can use.
```

X-kernelin asema Machin palvelimena merkitsee sitä, että merkittävä tietoliikennekomponentti ei olekaan käyttöjärjestelmän ytimessä, vaan ytimen ulkopuolella käyttäjän tilassa. Ytimessä on vain verkkokortin laiteohjain. Tällöin yksi fyysinen laitteisto voi mm. esiintyä useampana loogisena toisestaan riippumattomana järjestelmänä omine IP-osoitteineen. Kehityslaitteistolla olikin kaksi eri IP-osoitetta, toinen X-kernelille ja toinen Lites-kehitysympäristölle. Jostain selvittämättömäksi jääneestä teknisestä ongelmasta johtuen nämä eivät pystyneet keskustelemaan keskenään. Mahdolli-

sesti Machin laiteohjain ei osannut välittää koneesta lähteviä tietoliikennepaketteja takaisin samaan koneeseen.

Seuraavaksi oli tarkoituksena sijoittaa SNMP X-kernelin päälle protokollaksi, operaatio, jota X-kernel tukee erinomaisesti. Käytännössä tämä olisi kuitenkin edellyttänyt socket-rajapinnan ainakin osittaista toteuttamista.

Valitettavasti jouduin keskeyttämään projektin vuoden 1995 lopussa työtehtävien takia, eli SNMP:n ja X-kernelin yhdistäminen jäi tekemättä. X-kernelistä jäi vaikutelma hyvästä tutkimusplatformista.

4.7. Väliversio juoma-automaatista

Nixu Oy:n tupaantulijaisiin vuonna 1996 haluttiin saada toimiva juoma-automaatti. Tähän saakka ohjelmistokehitys oli suoritettu ilman virvoitusjuoma-automaattia, jota oli käytetty manuaalisesti ohjausjärjestelmää odotettaessa (rahastuslaitteisto oli korvattu ylimääräisellä painikkeella, ns. bonus-napilla). Nyt Lauri Aarnio korvasi automaatin kytkimet relekortin lähdöillä ja automaatin etulevyn tummalla pleksilasilla, jossa lukee "http://www.nixu.fi/limu/".

SNMP-osuus ei ollut valmistunut, joten Machin päälle kirjoitettua testiohjelmaa käytettiin pullojen pudottamiseen automaatista. Lauri Aarnio ja Timo Pekurinen kirjoittivat Perl-kielisen edustaohjelman, jolla oli oma tietokanta automaatin tilasta, ja joka otti rcp-protokollalla yhteyden automaattiin ja käynnisti pullojenpudotusohjelman. Johtuen Litesin ja Machin yhteensopivuusongelmista, kyseisestä ohjelmasta oli mahdotonta saada paluukoodia, joten automaattia ohjattiin sokkona.

Tähän päättyi projektin ensimmäinen osuus talvella 1996. Virvoitusjuoma-automaatti toimi Mach-käyttöjärjestelmän ohjauksessa, mutta toteutus oli karkea, eikä voida edes puhua varsinaisesta ohjausprotokollasta.

5. Toimivan prototyypin rakentaminen

5.1. Linux

Kun virvoitusjuoma-automaatin kehitys taas jatkui vuonna 1997, oli Machista ja Litesistä käytettävissä uudet versiot. En kuitenkaan saanut näitä kääntymään ja päätin lopulta, että koska Mach on jo tullut nähtyä, ja koska sen kehitys muualla (CMU:ssa) oli päättynyt ja huonosti tuetun ympäristön kanssa sähläämiseen kuluu vain voimia hukkaan, voisin unohtaa Machin. SNMP-ohjaus kuitenkin kiinnosti edelleenkin ja se haluttiin toteuttaa, vaikkapa sitten suositussa Linux-ympäristössä.

Uudeksi kehitysympäristöksi valittiin Machin tilalle Linux, joka oli tällä välin kehittynyt voimakkaasti ja joka on vapaasti jaeltavista Unix-yhteensopivista käyttöjärjestelmistä suosituin ja parhaiten tuettu. Muita vaihtoehtoja olisivat voineet olla esimerkiksi NetBSD ja FreeBSD.

Linux-käyttöjärjestelmän asentaminen työkaluineen sujui vaivattomasti. Linuxin Howto -dokumentaation [LinuxHowto] mukan Linux-järjestelmä voitaisiin tiivistää ROM-muistista tai kahdelta levykkeeltä käynnistyväksi. Tätä mahdollisuutta sulautetun järjestelmän toteuttamiseksi ei lähdetty tutkimaan.

Tässä yhteydessä en pitänyt enää tarpeellisena kirjoittaa I/O-kortille laiteohjainta, koska se ei olisi tuonut työhön mitään uutta. Machin laiteohjaimet ovat kuitenkin kohtuullisen lähellä Unixin laiteohjaimia. Sen sijaan I/O-korttia käytetään suoraan SNMP-agenttiohjelmistosta. Linux tarjoaa ioperm(2)-systeemikutsun, joilla saadaan käyttöoikeus x86-prosessorin I/O-osoitteisiin, sekä inb(2) ja outb(2)-systeemikutsut yksittäisten tavujen lukemiseen ja kirjoittamiseen. ioperm()-kutsu edellyttää pääkäyttäjän oikeuksia.

5.2. SNMP-toteutus

Virvoitusjuoma-automaatin SNMP-hallinnan toteuttaminen Linux-ympäristöön on huomattavasti helpompaa kuin Mach/X-kernel -ympäristöön. Linuxille ja muille Uni-

xieille on saatavilla useita valmiita SNMP-agentteja, joihin voidaan lisätä tarvittavat toiminnot.

5.2.1. Virvoitusjuoma-automaatin tietomalli

Päätin että virvoitusjuoma-automaatin etäkäyttämiseksi on saatava tieto automaatin nimistä, pulloista ja automaatin käytettävyydestä (juuri pudotetun pullon perään ei voi pudottaa toista, joten pullojen välillä on 30 sekunnin tauko). Lisäksi on jotenkin saatava automaatille käsky pudottaa haluttu pullo.

Virvoitusjuomapullot ovat automaatissa pystylinjoissa. Agentti tarjoaa SNMP:llä hallintaohjelmalle tiedon kussakin linjassa olevien pullojen nimikkeestä, pullojen määrästä ja kylmien pullojen määrästä. Tämä tieto on itse asiassa agentin käsitys automaatin tilasta, agentti ei tiedä mitä automaatissa todellisuudessa on. Tämä on varsin olennainen ajatus koko järjestelmän toiminnan kannalta. Agentti tarjoaa sopivan abstraktiotason näkymän automaatin toimintaan peittäen todellisen toteutuksen taakseen. Esimerkiksi automaatissa ei ole lämpömittaria vaan kylmien pullojen määrä arvioidaan täytöstä kuluneen ajan perusteella.

5.2.2. SNMP-agentin toteutus

Ilmaisjakelussa olevia SNMP-toteutuksia oli verkossa saatavilla useita [SimpleWeb]. Tarjoilla olevista vaihtoehdoista valitsin agentin toteutukseen UCD:n SNMP-toteutuksen, joka perustuu CMU:n SNMP-kirjastoon [UCD], lähinnä keskusteluryhmässä comp.protocols.snmp olleiden kommenttien perusteella. UCD:n paketti mainosti olevansa laajennettava agentti ja tämä pitikin paikkansa. Laajennettavuus on toteutettavissa jopa Unixin komentoja käyttäen, mutta tässä yhteydessä valitsin tehokkuussyistä C-kielisen ohjelmamodulin.

UCD:n paketin ohjeet olivat suunnilleen "kopioi esimerkkimoduli ja laajenna siitä omaksi MIB-toteutukseksi". Tämä onnistui kohtuullisen hyvin. Pieniä ongelmia tuli matkalla kuten se, että MIB-osoitteen pituus oli koodattu numerona itse koodin jouk-

koon ja se oli sieltä tunnistettava ja muutettava.

Toteutettu ohjelmamoduli käsittelee sille tulevat SNMP-viestit ja ylläpitää tietokantaa virvoitusjuoma-automaatin tilasta. Koska koko SNMP-agentti on toteutettu Unixin daemonina eli on muistissa pysyvä ohjelma, moduli ylläpitää tietokannan tilaa ohjelman muuttujissa sekä tallettaa sen levylle tekstitiedostoon. Agentti käynnistyy muiden Unix-daemonien tapaan koneen käynnistyessä

```
snmpwalk limu.nixu.fi public 1.3.6.1.4.1.1625.1

enterprises.1625.1.1.0 = "Nixu Oy:n juoma-automaatti"
enterprises.1625.1.2.1.1.1 = 1
enterprises.1625.1.2.1.1.2 = 2
enterprises.1625.1.2.1.1.3 = 3
enterprises.1625.1.2.1.1.4 = 4
enterprises.1625.1.2.1.2.1 = "Jaffalight"
enterprises.1625.1.2.1.2.2 = "Vichy Novelle"
enterprises.1625.1.2.1.2.3 = "Coca-Cola Light"
enterprises.1625.1.2.1.2.4 = "Coca-Cola"
enterprises.1625.1.2.1.3.1 = 5
enterprises.1625.1.2.1.3.2 = 20
enterprises.1625.1.2.1.3.3 = 18
enterprises.1625.1.2.1.3.4 = 4
enterprises.1625.1.2.1.4.1 = 5
enterprises.1625.1.2.1.4.2 = 7
enterprises.1625.1.2.1.4.3 = 6
enterprises.1625.1.2.1.4.4 = 4
enterprises.1625.1.3.0 = 0
enterprises.1625.1.4.0 = 0
```

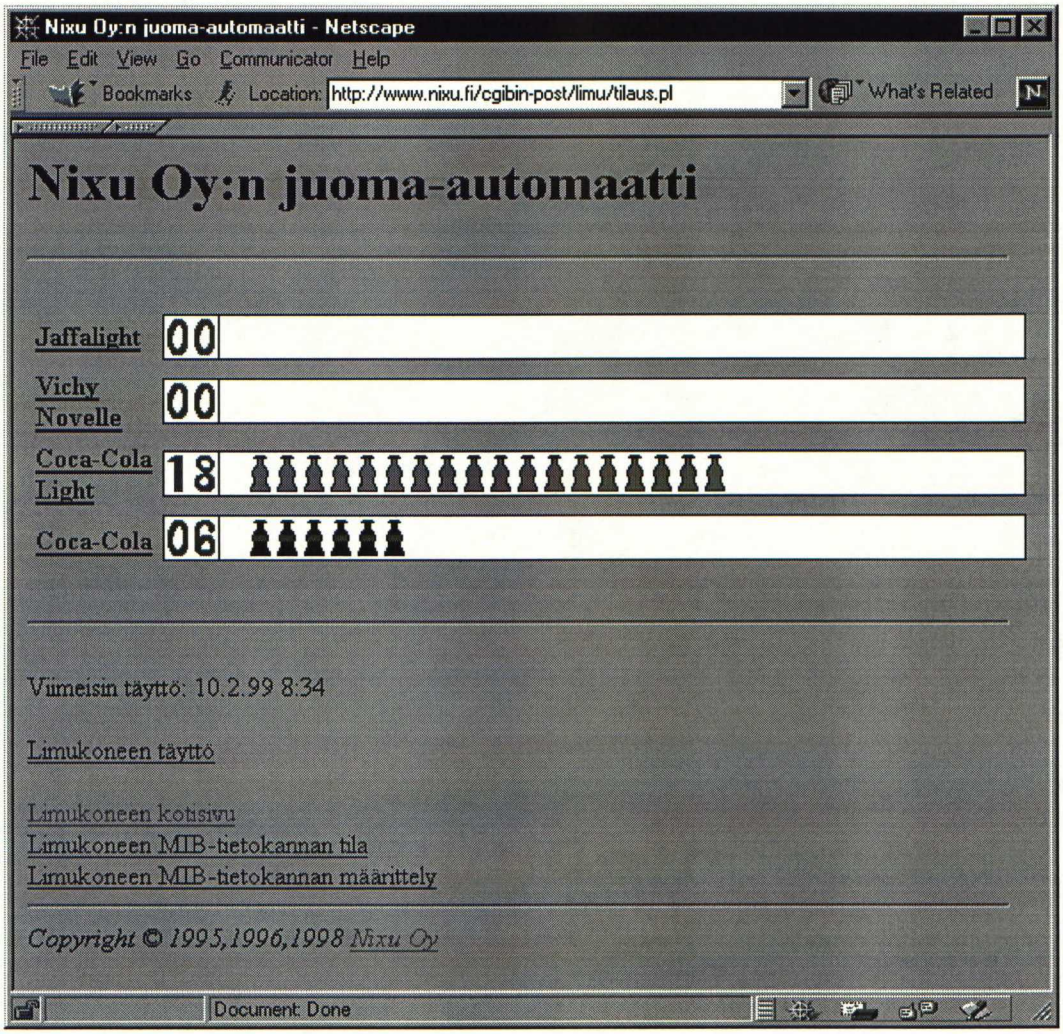
Kuva 8. Virvoitusjuoma-automaatin MIB-tietokannan sisältö, tietokannan määrittely on liitteessä 1

5.2.3. Käyttöliittymä

Lopullisessa muodossa virvoitusjuoma-automaatin tarjoama rajapinta on sen SNMP-liittymä. Projektin kuluessa automaattiin syntyi kaksi käyttöliittymää. Perl-kielisen WWW-käyttöliittymän rakensivat Timo Pekurinen ja Lauri Aarnio. Tämä käyttöliittymä on tyypillinen WWW-käyttöliittymä pienine pulloja esittävineen kuvineen. Huomattavaa on, että käyttöliittymä, joka on toiminnallisesti SNMP-hallinta-asema, ylläpitää omaa tietokantaansa, jossa säilytetään tietoa automaatin linjojen sisältöä

vastaavista pullojen kuvista. On konseptuaalisesti merkittävää, että sekä käyttöliittymä että virvoitusjuoma-automaatti ylläpitävät omaa tietokantaansa. Pulloja esittävät kuvakkeet eivät kuulu automaattiin vaan käyttöliittymään.

Projektin loppupuolella, kun virittelin SNMP:n vaatimia muutoksia WWW-käyttöliittymään, tein samalla triviaalin komentorivi- ja tekstipohjaisen käyttöliittymän automaattiin Perlillä. Lisäksi yrityksen sisällä on ollut kiinnostusta tietynmerkkisten juomien tilan aktiiviseen seurantaan ja oletettavissa on, että lähiaikoina näemme ihmisten rakentavan omia personalisoituja käyttöliittymiään laitteeseen. Yhteistä näille kaikille on yksinkertainen SNMP-rajapinta.



Kuva 9. WWW-käyttöliitymä juoma-automaattiin

```
jalopeno limu 127$ limu

          Nixu Oy:n juoma-automaatti

# Nimi               määrä lämpimiä
1 Jaffalight             0  0
2 Vichy Novelle          0  0
3 Coca-Cola Light       18  0
4 Coca-Cola              6  0
jalopeno limu 128$ _
```

Kuva 10. Tekstipohjainen SNMP-käyttöliittymä

5.3. Lopullinen virvoitusjuoma-automaatti

Tammikuussa 1999 virvoitusjuoma-automaatti toimii edellä kerrotussa muodossa SNMP-ohjauksessa. Osoitteessa http://www.nixu.fi/limu/ on automaatista kertova WWW-sivu, jonka kautta voi mm. lukea automaatin MIB:in tilan, virvoitusjuomien osalta. Tietoturvasyistä SNMP-liikennettä ei päästetä Internet-verkosta suoraan automaatille.

6. Uusin ohjausteknologia 1998

Koska työ oli venynyt ajallisesti suunniteltua pidemmäksi, tein vuonna 1998 uuden kirjallisuustutkimuksen. Merkittävin uusi kehitys on tällä välillä ollut Java-teknologioiden yleistyminen. Muutoin kehitys on ollut tavallista pienten askelten kehitystä.

6.1. Mach

Machia kehitettiin CMU:ssa 1985-1994, jonka jälkeen Machin kehitys jatkui Utahin yliopistossa. Viimeinen versio Utahista on keväältä 1996, jolloin Utah ilmoitti luopuvansa Machin kehityksestä ja siirtyvänsä tutkimaan käyttöjärjestelmiä Fluke-nimisen mikrokernelin pohjalta. Machin kehitys on jatkunut Mach-ytimen Linuxiin yhdistävän MkLinuxin ja GNU-projektin Hurd-käyttöjärjestelmän muodossa.

MkLinux näyttää keskittyvän Apple Macintosh-ympäristöön ja PowerPC-prosessoriin, oletettavasti siksi, että useimmat muut vapaata Unixia kehittävät projektit ovat keskittyneet Intelin prosessoreihin.

Gnu Hurd on nyt liittoutunut Linuxin Debian-jakelua rakentavan joukon kanssa. On mahdollista että Machin taru jatkuu. Hurd on sikäli merkittävä käyttöjärjestelmä, että sen kautta GNU-projekti voi saavuttaa alkuperäisen tavoitteensa, joka oli vapaasti lähdekoodimuodossa jaettava Unix-käyttöjärjestelmä. i386-arkkitehtuuriin sidotut Linux, NetBSD ja FreeBSD tosin ovat saavuttanet jo tämän tavoitteen käyttäen GNU-projektin tuottamia ohjelmia osana käyttöjärjestelmää.

Machin kohtalosta on vaikeaa sanoa mitään varmaa. Tällä hetkellä se näyttää palanneen alkuperäiseen tarkoitukseensa Unix-käyttöjärjestelmän ytimen pohjaksi.

6.2. Linux

Linux on kehittynyt voimakkaasti ja saavuttanut suuren kansainvälisen suosion palvelin- ja työasemakäyttöjärjestelmänä. Linuxia, kuten muitakin Unix-käyttöjärjestelmiä käytetään jonkin verran myös automaatiosovelluksiin, mutta mitään erityisen merkit-

tävää sijaa sillä ei alalla ole.

6.3. Windows NT

Microsoftin Windows NT on erityisesti palvelinkäytössä yleistynyt yleiskäyttöjärjestelmänä. Sitä käytetään myös jonkin verran sulautettuihin järjestelmiin. Kiinnostus Windows NT:n käyttämiseen sulautettujen järjestelmien käyttöjärjestelmänä ei johdu niinkään NT:n ominaisuuksista, vaan saatavilla olevista ja NT-ympäristön tuntevista ohjelmointiresursseista. NT:n tarjoama Win32 API lienee maailman käytetyin käyttöjärjestelmärajapinta, jolle on tarjoilla sekä osaavia ohjelmoijia että laadukkaita ohjelmankehitystyökaluja. [comp.realtime]

Windows NT:n omat reaaliaikaominaisuudet eivät tarjoa riittävää ennustettavuutta kovan reaaliaikaisuuden saavuttamiseen. Ristiriitaiset prioriteetit ratkaistaan epädeterminisesti, muistin lukitus ei ole täysin luotettava ja keskeytysten käsittely ei takaa minkäänlaisia vasteaikoja.

Reaaliaikalaajennuksia Windows NT:hen on saatavilla. Ne joko sijoittavat reaaliaikaiset prosessit NT:n ytimeen, jolloin ne ovat muistinhallinnan suojauksen ulottumattomissa, tai laajennukset käyttävät Intelin prosessorien erityisominaisuuksia NT:n hallintaan ja HAL-kerroksen (Hardware Abstraction Layer) keskeyttämiseen, jolloin reaaliaikainen prosessi voidaan suorittaa prosessorin käyttäjätilassa muistinhallinnan suojaamana.

6.4. Java-teknologia

Java on Sun Microsystemsin kehittämä teknologiaperhe, joka suunniteltiin alun perin sulautettujen järjestelmien ohjelmointiin, mutta joka on saavuttanut suosiota myös WWW-sivuille sijoitettavien sovellusten eli applettien ohjelmoinnissa. Olio-orientoituneella Java-kielellä kirjoitettu ohjelma käännetään laiteriippumattomalle välikielelle ns. tavukoodeiksi, jotka Java-tulkki suorittaa kohdekoneessa. Ohjelmointikielenä Java on huomattavasti ohjelmoijaystävällisempi kuin esimerkiksi koneläheinen C, jota on kutsuttu myös symboliseksi makroassembleriksi. Kielestä on poistettu useita

C-kielen laitteistoläheisyyden aiheuttamista ongelmista, esimerkiksi muuttujien pituus ei riipu prosessoriarkkitehtuurista ja suorat muistiosoitukset on estetty. Java hoitaa muistinhallinnan itse, eikä käyttäjän tarvitse puuttua siihen kuten C-kielessä. Hintana on kuitenkin se, että Java-kielinen koodi käännetään välikielelle, jota tulkataan suoritettaessa. Tämä ja erityisesti tulkin toteuttama dynaaminen muistinhallinta aiheuttaa ongelmia reaaliaikaisuuteen pyrittäessä.

Javan ansiosta on myös sulautettujen järjestelmien kohdalla on noussut esille termi vahvasti tyypitetty tai tyyppiturvallinen ohjelmointikieli. Tämä viittaa mahdollisten ohjelmointivirheiden vaikutukseen järjestelmän toimintaan. Esimerkiksi heikosti tyypitetyssä C-kielessä olevat osoittimet mahdollistavat osoituksen mihin tahansa muistin kohtaan ja riippuen allaolevasta muistinhallinnasta, ohjelma voi sotkea oman datamuistinsa, oman ohjelmakoodinsa tai koko järjestelmän muistin. C-kieltä ei täten voida pitää tyyppiturvallisena. Vahvasti tyypitetyssä Java-kielessä puolestaan ei ole osoittimia eikä suoraa muistinosoitusta, vaan ohjelmoija voi käsitellä ainoastaan nimettyjä muistiobjekteja, joihin on viitattava käyttäen oikeaa muuttujatyyppiä. Lisäksi Java-kielen ominaisuuksiin kuuluu virheiden käsittely; ohjelma voi selvitä itse omista virheistään. Java-kieli tukee täten vikasietoista ohjelmointia paremmin kuin C-kieli. Ajonaikainen tyypintarkistus tosin heikentää suorituskykyä. [Pfister98]

6.4.1. JavaOS ja Embedded Java

JavaOS on Java-kielen ominaisuuksia tukeva mikrokernelikäyttöjärjestelmä, joka soveltuu sulautettuihin järjestelmiin, jotka eivät vaadi kovaa reaaliaikaisuutta. Mikrokerneli tukee keskeytyksien käsittelyä ja säikeitä, mutta ei sisällä muistin tai säikeen lukitusta.

EmbeddedJava on uusi sulautettuihin järjestelmiin optimoitu ympäristö ja API-rajapinta Java-ohjelmille. Embedded Java tarvitsee alleen käyttöjärjestelmän, joka tarjoaa reaaliaikaisuuden, muistinhallinnan, ja tuen säikeille. Lisäksi käyttöjärjestelmä voi tarjota tiedostojärjestelmän, graafisen näyttökirjaston ja verkkopalvelut.

Tarkoituksena on, että alla olevan käyttöjärjestelmän toimittaja tai laitevalmistajat kirjoittaisivat laiteohjaimet laite-spesifisten luokkien puitteissa.

EmbeddedJava on suunnattu erityisesti sulautettujen järjestelmien ohjelmistokehitykseen. Se toteuttaa Java-kielen normaaliluokat muistia säästävässä muodossa, sekä sisältää työkalut luodun ohjelmiston muistintarpeen minimointiin ja koodin muuntamiseen ROM-muistiin sopivaan muotoon.

JavaOS ja Embedded Java ovat Sun Microsystemsin kaupallisia tuotteita, ja siten eivät QNX:n tavoin sovi alkuperäiseen tehtävän määrittelyyn. Muutoin JavaOS vaikuttaa potentiaaliselta käyttöjärjestelmältä tilanteisiin, joissa reaaliaikaisuutta ei vaadita. EmbeddedJava vaatii alleen reaaliaikaisen käyttöjärjestelmän ja sitä on katsottava pikemminkin reaaliaikaisen sulautetun järjestelmän ohjelmiston fyysisestä toteutuksesta erottavana rajapintana. Tulevaisuus näyttää kuinka paljon merkitystä tällä on. Mahdollisuus kehittää sulautettuja ohjelmistoja ja uudelleenkäyttää ohjelmakoodia eri laiteympäristöissä kiinnostaa kuitenkin ensisijaisesti niitä, jotka toteuttavat sulautettuja ohjelmistoja eri arkkitehtuureille, eivätkä nämä markkinat ole kovinkaan suuret. Tosin sulautettujen verkkoon liitettyjen järjestelmien markkinat ovat ylipäänsä vasta alussa.

6.4.2. Jbed

JBed on sveitsiläisen Oberon Microsystemsin kehittämä sulautettujen järjestelmien käyttöjärjestelmä, jossa on reaaliaikainen Java-virtuaalikone ytimeen täysin integroituna. JBed tukee sekä kovaa reaaliaikaisuutta vaativien sovellusten että laiteohjaimien kirjoittamisen Java-kielellä. JBed on toteutettu 68k ja Power-PC -arkkitehtuureille. Käyttöjärjestelmän muistitarpeet ovat satojen kilotavujen suuruusluokkaa.

Pelkästään näiden ominaisuuksien ansiosta JBed olisi kiinnostava käyttöjärjestelmä, mutta lisäksi se tuo kaksi uutta näkemystä sulautettujen reaaliaikaisten käyttöjärjestelmien maailmaan: muistinsuojauksen tarpeettomuuden ja prioriteetteihin perustumattoman skeduloinnin.

JBed white paper esittää kiintoisan perinteisestä ajattelusta eroavan katsantokannan luotettavuuteen. Perinteisesti luotettavuus on saavutettu erottamalla prosessit vahvasti toisistaan ja suojaamalla kukin prosessi muistinhallinnan avulla. Kommunikointi tällaisten prosessien välillä on kallista, ts. vie paljon keskusyksikköaikaa. Lisäksi prosessien välinen kommunikointi perustuu yleensä datatavujen siirtämiseen eikä hyödynnä oliomallin korkeamman tason abstraktioita.

JBed edellyttää että ohjelmointikielen kääntäjä ja ajonaikainen järjestelmä vastaavat oliomallin toteutumisesta. Kaikki muistin käsittely suoritetaan olioiden kautta, suoraa muistinosoitusta ei sallitta. Tällöin kaikki ohjelmistokomponentit voivat jakaa saman muistialueen.

JBed on selkeästi liian uusi järjestelmä, jotta siitä olisi riittävästi kokemuksia syvällisempää evaluointia varten. [Pfister98] tuo kuitenkin esille useita kiinnostavia näkökulmia, jotka ovat pohtimisen arvoisia.

Toisin kuin useimmissa muissa käyttöjärjestelmissä, JBedin lähtökohta ohjelmointivirheiden vaikutuksen rajoittamiseen ei ole prosessien eristäminen toisistaan muistinhallinnalla, vaan käytettyjen ohjelmointikielten vahva tyypitys ja sen valvominen ohjelmaa käännettäessä ja suoritettaessa. Tämä mahdollistaa prosessien korvaamisen samassa prosessoritilassa ajettavilla säikeillä ja nopeuttaa kontekstinvaihtoa ja tehtävien välistä kommunikointia huomattavasti. Vahvan tyypityksen on oltava todellakin vahvaa, C, C++, Modula-2 tai Ada eivät tarjoa riittävää suojaa. Java sen sijaan on aidosti vahvasti tyypitetty kieli, jossa normaalien luokkien puitteissa ei ole mitään keinoa osoittaa muistia suoraan tai pakottaa muuttujan tyyppiä toiseksi.

Reaaliaikaisuutta tavoitellessa vahvasti tyypitetyn oliokielen selkein ongelma on roskienkeruussa. JBedissä tämä on ilmeisesti ratkaistu, mutta lähteestä ei selviä miten.

Säikeiden skedulointiin JBed tarjoaa myös uuden paradigman. JBed poistaa prioriteettiabstraktion ja toteuttaa sen sijaan "Earliest Deadline First" -skedulerin, joka arvioi minkä säikeen absoluuttinen valmistumisajankohta on lähimpänä ja skeduloi

sen seuraavaksi. Prioriteettihan on reaaliaikajärjestelmässä sikäli väärä abstraktio, että normaalisti kyse ei ole tehtävien tärkeysjärjestyksestä, vaan kaikki tehtävät on suoritettava omien vaatimuksiensa mukaisesti. Skeduleri, joka käyttää mittayksikkönä mikrosekuntia, kuten JBedin skeduleri, on tässä mielessä mielekkäämpi kuin prioriteetteihin perustuva skedulointi.

Oberon väittää, että ominaisuuksiensa ansiosta JBed ei käytä muistia eikä keskusyksikköaikaa enempää kuin perinteiset reaaliaikaiset käyttöjärjestelmät.

6.5. SNMP

SNMP-protokollan ensimmäinen versio (SNMPv1), jota tässäkin työssä on käytetty, on kärsinyt ominaisuuksien puutteesta ja yksinkertaisuudestaan. Suurimmat ongelmat ovat olleet tarve hakea jokainen tietoalkio omalla kutsullaan ja tietoturvan puute. Tätä yritettiin korjata SNMPv2:ssa, mutta standardointi ei koskaan päässyt lopulliseen tulokseen ja SNMPv2:sta on kolme eri versiota. SNMPv3 on yritys koota standardi taas yhteen ja saada siihen tarvittavat ominaisuudet mukaan, mutta sekään ei ole vielä valmistunut. SNMPv3:a odotellessa SNMPv1:n tietoturvapuutteet voidaan ratkaista myös käyttämällä IPSEC-standardia, joka mahdollistaa kaiken IP-pohjaisen liikenteen salaamisen kahden koneen välillä sekä osapuolten kryptografisesti vahvan tunnistamisen.

6.6. Kenttäväylät

Kenttäväylätuotteet ovat kehittyneet ja yleistyneet ja kenttäväylät saavuttavat hitaasti mutta tasaisesti suosiota teollisuuden piirissä. Useimmille kenttäväylille on saatavilla tuotteita, jotka tunneloivat kenttäväylän liikenteen TCP/IP-protokollien avulla. Liitettäessä kenttäväyliä esimerkiksi Internetin välityksellä menetetään kuitenkin osa kenttäväylän ominaisuuksista, lähinnä reaaliaikaisuus.

6.7. WWW

World Wide Web on yleistynyt erilaisten järjestelmien käyttöliittymänä ja markki-

noille on ilmaantunut automaatiolaitteita, joita hallitaan WWW-käyttöliittymällä. WWW sopii tiedonsiirtoformaattina kuitenkin paremmin tiedon välittämiseen ihmisille kuin ohjelmistojen väliseen kommunikointiin.

Lisäksi WWW-järjestelmä tukee lukuisia laajennuksia, kuten Java-appletteja, joiden avulla WWW-sivulle voidaan sijoittaa ohjelmakomponentti. Java-teknologialla onkin jo toteutettu SNMP-hallinta-asema. Tällöin WWW-selainohjelma tarjoaa laitteistoriippumattoman käyttöliittymän, mutta varsinainen etähallinnan tiedonsiirto suoritetaan SNMP:llä.

7. Tulosten arviointi

Ajatus tähän työhön lähti tuotantoautomaatioon liittyvistä opinnoistani. Tietotekniikan opinnoissani olin nähnyt tietoliikenteen ja modernien ohjelmistonkehitysjärjestelmien kehityksen, mutta automaatiopuolella en nähnyt vastaavaa kehitystä. Syksyllä 1994 tuotantoautomaatio näytti edelleenkin paljolti pyörivän 8-bittisten pienimuististen mikroprosessorien ja releitä emuloivien ohjelmoitavien logiikoiden varassa ja tietoliikenne, jos sellaista oli, ei noudattanut muita kuin matalimman tason julkisia standardeja.

Työn alussa suorittamani kirjallisuustutkimus vahvisti käsitystäni. Työn aikana uskomukseni alkoi kuitenkin horjua ja työn lopussa suorittamani toinen kirjallisuustutkimus alkoi vakuuttaa minua siitä, että vaikka suuri osa erilaista automaatiota edelleenkin perustuu tekniikoihin, joita voidaan pitää vanhentuneina, on tämä usein perusteltua. Automaatioala on tietoinen tietotekniikan kehityksestä, mutta soveltaa uusia ideoita usein konservatiivisesti. Tuotannon luotettavuusvaatimukset ovat toista luokkaa kuin esimerkiksi toimistotietojenkäsittelyssä.

Automaatioala voisi kuitenkin hyötyä muun tietotekniikan kehityksestä, esim. kenttäväylien standardoinnissa pohditaan osoitekäytäntöjen kohdalla ongelmia, jotka on jo ratkaistu verkonhallinnan puolella.

7.1. Tavoitteiden saavuttaminen

Työhön ryhtyessäni ajatuksenani oli kehittää ja kuvata moderni ympäristö sulautettujen järjestelmien toteutukseen. Ympäristö, jossa ohjelmankehitys suoritettaisiin moderneilla työkaluilla kehittyneessä ja tehokkaassa ympäristössä ja joka tukisi globaalin mittakaavan tietoliikennettä riittävän korkealla abstraktiotasolla. Tämä tavoite saavutettiin osittain, mutta ei täysin.

Mach ei osoittautunut niin laadukkaaksi ympäristöksi kuin alustavan kirjallisuustutkimuksen perusteella olin kuvitellut. Vuonna 1995 Mach vaikutti lupaavalta tulevaisuu-

den käyttöjärjestelmältä. Vuonna 1998 vaikuttaa siltä, että vaikka Machin ympärillä tapahtuukin jotain pienimuotoista tutkimusta, se on käytännössä liki kuollut. Sulautettuihin järjestelmiin se olisi tuskin muutenkaan soveltunut suuren kokonsa (n. 1 Mtavu) takia.

Tietoliikennepuolella työn anti oli runsaampaa. Tehdyn tutkimuksen perusteella vaikuttaa siltä, että automaatioalalta puuttuvat etävalvontaan ja -ohjaukseen tarvittavat korkeamman tason tietoliikennestandardit. Nykyiset ratkaisut ovat vielä matalan tason standardeja puhtaan anturointidatan siirtämiseen, ollen usein vielä valmistajakohtaisia suljettuja standardeja. SNMP:n tapaiselle korkean abstraktiotason protokollalla on nähtävissä selkeä tarve. Toteutettu virvoitusjuomakoneen ohjaus useine käyttöliittyminen (WWW, komentorivipohjainen, verkonvalvontaohjelmisto) osoittaa, että SNMP-protokollaa voidaan käyttää myös fyysisten laitteiden ohjaukseen.

Työssä toteutettu järjestelmä ei Linux-pohjaisena kuitenkaan ole kovinkaan käyttökelpoinen reaaliaikaisten prosessien ohjaukseen tai muutoinkaan sulautettujen järjestelmien toteutukseen. Luotettavuus ei ole ohjauslaitteelta vaadittavalla tasolla. Järjestelmä ei myöskään ole kustannuksiltaan riittävän halpa laajalti levitettäväksi. Pidemmällä aikavälillä tekniikan kehittyessä ja Internet-verkon levitessä kaikkialle tässä esitetty tekniikka on kuitenkin sovellettavissa myös käytännön vaatimuksiin.

7.2. Mitä toteutetusta prototyypistä opittiin

Machia kääntäessäni ja X-kerneliä Machille siirtäessäni opin paljon käyttöjärjestelmistä ja niiden sisäisestä toiminnasta. Itse asiassa enemmän kuin olisin halunnut, käyttöjärjestelmien tekniset ongelmat veivät projektista suhteessa kohtuuttomasti aikaa. Operaatio oli kuitenkin hyvin opettavainen, koska en ollut aikaisemmin penkonut syvällisemmin Unixin kirjastojen ja käännösympäristön käyttäytymistä.

Tehdyn työn perusteella arvioin että Mach ei ole hyvä ympäristö sulautettujen järjestelmien toteuttamiseen. Tutkimuskäyttöön se saattaa sopia hyvinkin. Jokin kaupallinen ja hyvin tuettu sulautettujen järjestelmien käyttöjärjestelmä, kuten QNX, olisi

ollut varmastikin taloudellisesti kannattavampi.

SNMP-ohjauksen toteutus oli yllättävän helppoa, vaikka senkin yhteydessä törmättiin pieniin ongelmiin. SNMP-protokolla on valitettavan vähän tunnettu ja hyödynnetty. Yksi syy tähän lienee alan kirjallisuus, joka tyypillisesti sukeltaa turhan nopeasti formaaleihin määrittelyihin ja muuhun teoriaan. Itse totesin työtä tehdessäni, että SNMP on tosiaankin varsin yksinkertainen ymmärtää, joskin ASN.1-koodatut MIB-määrittelyt aiheuttavat ylimääräistä työtä. SNMP:n rakenne on itse asiassa yksinkertaisempi kuin yleensä annetaan ymmärtää.

7.3. Miten työ tehtäisiin nyt?

Vuonna 1999 horisontissa häämöttävät Java-teknologiaan perustuvat sulautetut järjestelmät, joissa oliomallin toteutuminen vahvassa muodossa muuttaa perinteistä näkemystä käyttöjärjestelmistä. Tämä teknologia olisi uuden tutkimuksen aihe.

Tietoliikennepuolella SNMP vaikuttaa edelleenkin varteen otettavalta ratkaisulta erilaisiin etävalvonta- ja ohjaussovelluksiin. Hiljalleen standardoituvat kenttäväylät ovat erinomainen ratkaisu paikallisiksi pienen tiedonsiirtovolyymin anturointiväyliksi, mutta laaja-alaisempaan käyttöön niistä ei ole. SNMP on globaali protokolla, joka tarjoaa sopivan abstraktin ja laitteistoriippumattoman näkökulman ohjattavaan laitteistoon.

SNMP voisi soveltua myös kenttäväylässä siirrettäväksi abstraktiokerroksen tarjoavaksi laitteenkuvausprotokollaksi, SNMP:n sidonta TCP/IP-protokolliin ei ole mitenkään ehdoton. Käytännössä kuitenkin tämäntapaisten ajatusten läpi saaminen kenttäväyläpuolella lienee mahdotonta.

Jos tekisin työn nyt uudestaan, valitsisin sulautetun järjestelmän käyttöjärjestelmäksi kaupallisen tuotteen, kuten QNX:n ja sen päälle toteutusympäristöksi jonkin Java-ympäristön. Tietoliikenteen toteuttaisin SNMP:llä ja TCP/IP:llä kuten oli suunnitelma alunperinkin.

8. Lähteet

8.1. SNMP, verkonhallinta ja tietoliikenne

[Aarnio94] Aarnio, L. "Kassapäätteen perusohjelmisto paikallisverkkoympäristössä", diplomityö, Teknillinen Korkeakoulu 1997

[RFC1149] D. Waitzman, "A Standard for the Transmission of IP Datagrams on Avian Carriers", Request for Comments 1149, 1990

[RFC 1157] Case, J., Fedor, M., Schoffstall, M. & Davin, J., "A Simple Network Managment Protocol", Request for Comments 1157, 1990

[RFC 1158] Rose, M. (editor), "Management Information Base Network Management of TCP/IP based internets: MIB-II", Request for Comments 1213, 1990

[RFC1213] Rose, M. (editor), "Management Information Base for Network Management of TCP/IP based internets: MIB-II", Request for Comments 1213, 1991

[Rose96] Rose, M., "The Simple Book: an Introduction to Network Management", Prentice Hall, 1996

[SimpleWeb] "The Simple Web", verkkojulkaisu osoitteessa http://wwwsnmp.cs.utwente.nl/

[Stallings96] Stallings, W., "SNMP, SNMPv2, and RMON: Practical Network Management - 2nd ed.", Addison-Wesley Publishing Company, 1996

[UCD] UCD:n SNMP-toteutus osoitteesta: ftp://ftp.ece.ucdavis.edu/pub/snmp/ucd-snmp.tar.gz

8.2. Mach ja muut käyttöjärjestelmät

[Boykin93] Boykin, J., Kirschen, D., Langerman, A. & LoVerso, S., "Programming under Mach", Addison Wesley, 1993

[comp.os.mach95] verkkokeskusteluja comp.os.mach -ryhmässä 1995

[comp.realtime98] verkkokeskusteluja comp.realtime -ryhmässä syksyllä 1998

[comp.realtime FAQ] comp.realtime -keskusteluryhmän usein kysyttyjen kysymysten lista, saatavilla osoittteessa http://www.faqs.org/faqs/realtime-computing/faq/

[LinuxHowto] Linux-käyttöjärjestelmän dokumentaatio, saatavilla osoitteessa http://www.linux.org/help/howto.html

[Pfister98] Pfister, C., "JBed Whitepaper: Component Software and Real-Time Computing", Oberon Microsystems, 1998, saatavilla osoitteessa http://www.oberon.ch/rtos/whitepaper/

[RTMAG] "Real-Time Magazine" -verkkojulkaisu, osoitteessa http://www.realtime-info.be/

[Tanenbaum92] Tanenbaum, A., "Modern Operating Systems", Prentice-Hall, 1992

8.3. Automaatiotekniikka ja logistiikkka

[Aaltonen92] Aaltonen, Andersin, Airila, Ekman, Kauppinen, Liukko, Pohjala, "Tuotantoautomaatio", Otatieto, 1992

[Nyberg95] Keskusteluja Timo R. Nybergin (Teknillinen korkeakoulu, koneosasto) kanssa 1995

[Paavilainen97] Liisa Paavilainen., "Kiinteistöjen tiedonsiirron kehitysnäkymät", diplomityö, Teknillinen Korkeakoulu 1997

[Pekurinen 94] Keskustelu Timo Pekurisen kanssa virvoitusjuoma-automaatin logiikan käyttöjännitteen tason empiirisestä havaitsemisesta syksyllä 1994

[Pyyskänen95] Pyyskänen S., Hyvärinen J., "Liike ja tuotantokiinteistöjen tiedonsiir-

ron kehittäminen", duocon-electro oy 1995

[Sahlstén99] Keskusteluja Toivo Sahlsténin (Helsingin kaupungin Rakennusvirasto) kanssa joulukuussa 1998 ja tammikuussa 1999

[Sakki94] Jouni Sakki, "Logistinen materiaalin ohjaus", MH-Konsultit Oy 1994

Liitteet

Liite 1: Virvoitusjuoma-automaatin MIB-määrittely

```
LIMU-MIB DEFINITIONS ::= BEGIN

IMPORTS
        enterprises, OBJECT-TYPE
    FROM SNMPv2-SMI
DisplayString
    FROM SNMPv2-TC;

nixu OBJECT IDENTIFIER ::= { enterprises  1625 }
limu OBJECT IDENTIFIER ::= { nixu  1 }

limuBanner OBJECT-TYPE
    SYNTAX      DisplayString
    ACCESS      read-write
    STATUS      current
    DESCRIPTION
        "Banner to be displayed by the management station."
    ::= { limu  1 }

LimuLineIndex ::= INTEGER

limuLineTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LimuLineEntry
    ACCESS      not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing the information of bottles in the lines"
    ::= { limu  2 }

limuLineEntry OBJECT-TYPE
    SYNTAX      LimuLineEntry
    MAX-ACCESS not-accessible
    STATUS      current
    DESCRIPTION
        "A conceptual row of the limuLineTable containing the status
         of a particular line.  Each row of this table is transient."
    INDEX       { limuLineIndex }
    ::= { limuLineTable  1 }


LimuLineEntry ::= SEQUENCE
        { limuLineIndex      INTEGER,
          limuLineName        DisplayString,
          limuLineBottles     INTEGER,
          limuLineColdBottles INTEGER}
```

```
limuLineIndex OBJECT-TYPE
    SYNTAX      LimuLineIndex
    MAX-ACCESS read-only
    STATUS      current
    DESCRIPTION
        "A unique value, greater than zero, for each
         line of bottles.  It is recommended that values are assigned
         contiguously starting from 1."
    ::= { limuLineEntry  1 }

limuLineName OBJECT-TYPE
    SYNTAX      DisplayString
    ACCESS      read-write
    STATUS      current
    DESCRIPTION
        "The name of this line to be displayed, usually the
         brand name."
    ::= { limuLineEntry  2 }

limuLineBottles OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      current
    DESCRIPTION
        "Amount of bottles available on this line."
    ::= { limuLineEntry 3 }

limuLineColdBottles OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      current
    DESCRIPTION
        "Amount of cold bottles available on this line."
    ::= { limuLineEntry 4 }

limuBusy OBJECT-TYPE
    SYNTAX      INTEGER (0..'ffff'H)
    ACCESS      read-only
    STATUS      current
    DESCRIPTION
        "If 0, machine will accept requests, otherwise
         seconds left until macine is usable."
    ::= { limu  3 }

limuEject OBJECT-TYPE
    SYNTAX      INTEGER (0..'ffff'H)
    ACCESS      read-write
    STATUS      current
    DESCRIPTION
        "Request a bottle to be ejected from numbered line.
         Return values:
           0 = not succesfull
```

```
            n = (n > 0) ejecting on line n"
    ::= { limu  4 }

END
```